Visual Navigation on Rough Tracks in Bush Environments

David Fernandez
Bachelor of Computer Science and Engineering (with Honours) (2001)

Submitted for the Degree of
Doctor of Philosophy

Intelligent Robotics Research Centre,
Department of Electrical & Computer Systems Engineering,
Monash University, Clayton, VIC 3800, Australia

May 2007

Contents

List of Figures

List of Algorithms

Abstract

Natural environments can be very challenging workspaces for humans and machines. Unlike urban environments that are designed, built and maintained by man, natural environments can appear random and arbitrary, with little or none of the structure, regularity or consistency that so often makes even the simplest tasks achievable. Bush environments are essentially natural environments, though often some structure does exist in the form of the many rough tracks – often referred to simply as dirt roads – that have been cut into the landscape to allow easier access, sometimes for bushfire control, for example. These environments, typified by State and National Parks that cover much of Australia's south-east regions, present a worthwhile challenge to the field of robotics. These areas are the common workplace of a number of important groups, notably fire-fighters, search and rescue personnel and park and forest management personnel from Australian departments and authorities. Around the globe, similar local agencies – such as the California Department of Forestry and Fire Protection in the United States – perform largely the same duties in very similar environments. These people work in a difficult environment, often in time-critical and life-threatening operations. With a very small number of people trying to manage a very large area of land, help in the form of autonomous or semi-autonomous robots could be of great benefit.

The work presented in this thesis aims to address the problem of how navigation may be performed in bush environments, specifically navigation along poorly structured dirt roads. To be efficient (in terms of say energy consumption or mission time) and to give the highest chance of mission success, navigation *should* make use of roads whenever possible, as they are far more likely to offer safe and unhindered and hence rapid travel than the alternative of trekking cross-country. Because of the irregularity, inconsistency and low level of structure exhibited by dirt roads, traditional urban road-following techniques that make use of painted road markings and consistent road geometry are simply not applicable. Instead, alternative methods must be employed that can handle these and other uncontrollable environmental conditions. With the budgets of relevant agencies often stretched, cost is an important factor, as is mechanical simplicity. In balancing these requirements against adequate sensing requirements, the choice was made to use passive vision only, that is, only simple, cheap and commonly available cameras are to be used to provide the necessary input, coupled with standard computing equipment to perform processing. Comparatively expensive and often complex sensors using lasers, sonar or active vision are avoided.

The navigation system presented is separated into a number of components:

Visual Odometry A visual odometry system provides a measure of the robot's motion in a way that allows both intended (that is, driven) motion and accidental motion (such as that due to slippage) to be measured. Ground-facing cameras and optical flow techniques are employed to estimate the motion of the ground relative to the robot and

hence allow the robot's motion to be recovered.

Obstacle Detection A stereo-system provides an obstacle likelihood map that describes the visible environment in terms of the likelihood of a region being an obstacle, based upon its variation from a locally estimated flat ground plane. In this manner, both positive obstacles (such as fallen branches) and negative obstacles (such as potholes) can be identified and hence avoided. By performing local ground-plane estimation, the system can adapt to changes in the terrain that, while posing no threat, could otherwise be falsely considered unsafe.

Road Detection A colour-based road-finder produces a road-likelihood map that gives the likelihood of a visible region being part of the road. A set of locally-sampled statistical colour descriptors is used that accommodates changes in the appearance of the road surface due, for example, to changes in road material or variations in lighting. An intelligently adaptive set of colour filters (a short-term "memory" of the road surface) accommodates changes in the road surface while minimising redundant information.

Local Path-planning A local path-planner combines the two likelihood maps to form an overall amenability map which shows regions that are believed to be both free of obstacles and part of the road. It can also simply ignore road information if and when cross-country traversal is required. By applying some simple geometric assumptions a path is produced that aims to provide the robot with a safe and smooth trajectory. A number of factors, including path length, average width, smoothness and relative bearing are used to give an overall path fitness which is then used to determine the best path.

Global Path-planning Finally, a global path-planner uses a predefined, hybrid metric-topological map to find a route from start location to desired destination that makes use of roads when possible but allows for the generation of intermediate straight-line, cross-country paths if beneficial. The map consists of nodes – geographically distinct "places-of-interest", typically corresponding to intersections and intermediate waypoints – and edges – connecting roads – that are determined manually *a priori* from a standard map. In planning a path between given source and destination nodes, roads (real edges) are favoured over cross-country traversal (*pseudo*-edges), though cross-country trekking is allowed if the perceived distance benefits outweigh the anticipated reduction in safety.

Declaration

I declare that:

1. This thesis contains no material that has been accepted for the award of any other degree or diploma in any university or institution, and

2. To the best of the my knowledge and belief, it contains no material previously published or written by another person, except when due reference is made in the text of the thesis.

.....
(Signed) **David Fernandez**

.....
Date

Acknowledgements

I would like to thank and acknowledge my main supervisor Professor Ray Jarvis who made it possible for me to turn my ideas into a worthwhile piece of research. His wealth of knowledge and a desire to share it, his generosity with resources and his commitment to ensuring he is always available for consultation and discussion are all greatly appreciated. For his input and support during the early stages of my work and for sharing his enthusiasm for robotics, I would like to thank Andrew Price. I would also like to acknowledge and give thanks for the support and assistance given by the support staff within the Department of Electrical and Computer Systems Engineering, in particular Tony Brosinski and Maurice Gay in the Mechanical Workshop and Ray Cooper.

To my friends, both in and out of the lab, I am thankful. To list every one and all they had done for me would require a second volume, so I will (unfortunately) have to keep this short. I consider myself highly fortunate to have shared laboratory and office space with a truly remarkable group. The label *colleagues* does not do them justice: they are all true friends. Gideon Kowadlo's seemingly endless supply of energy and enthusiasm (and coffee, which may or may not be related) kept my spirits high no matter how far away completion seemed. Robert Stewart's at times crazy ideas have been a constant reminder of why I pursued this line of research and why it will always hold my interest. Dorian Spero taught me to have faith in your abilities and your ideas, even if they *don't* involve the use of a Kalman Filter and perhaps even moreso when they don't. Dave Rawlinson's knowledge of high-performance Japanese sports cars has provided many hours of discussion and questioning that, while not strictly work-related, kept my intellect stimulated and my mental cogs moving. Jay Chakravarty has been a source of infectious laughter and the lightbulb above his head (or sometimes taped to it) has produced many a stimulating discussion. Steve Woon's blocking moves on a basketball court made it clear that things don't always go your way, but that that's no reason to stop trying. Leon Cui and his love of the outdoors and off-road vehicles, which in part lead to events which inspired some of my ideas, has provided a source of entertainment and relaxation outside the lab that was at times much-needed. Crystal Barnes, despite being thousands of miles away, has in times of need been a rock. Her love and support have been unwavering and I am forever indebted to her. She also turned me on to *SpongeBob SquarePants* and I am glad that she did. Finally, my dearest kitty, Lara Kornienko. While a late entrant, she has given her shoulder and her ear, as well as her heart, and is a constant reminder that determination can triumph over adversity.

Finally, and most importantly, I thank my family. I am forever grateful to my mother Elizabeth, father Geoffrey and sister Christine for their love and support throughout the ups and downs of my research. My gratitude is also given to my other "sibling": Bronson. This big, drooling, furry bundle brought me countless hours of fun and happiness during what were on occasion difficult times and I miss him dearly. All have been constant sources of comfort and stability and I thank them sincerely with all of my heart.

This research was gratefully supported by *Australian Postgraduate Award* (Australian Government Department of Education, Science and Training) and *Information & Communications Technologies* (Victorian Department of Education and Training) scholarships.

List of Acronyms

DGPS	Differential Global Positioning System See also GPS
GPS	Global Positioning System
HSI	Hue, Saturation, Intensity
HSL	Hue, Saturation, Lightness
HSV	Hue, Saturation, Value
INS	Inertial Navigation System
ISO	International Organization for Standardization
LIDAR	Light Detection and Ranging
RADAR	Radio Detection and Ranging
RANSAC	Random Sample Consensus
RGB	Red, Green, Blue
SLAM	Simultaneous Localisation and Mapping
SVD	Singular Value Decomposition
USAR	Urban Search and Rescue
USB	Universal Serial Bus
WAAS	Wide Area Augmentation System
XYI	X, Y, Intensity

Chapter 1

Introduction

> “ *Rose is a rose is a rose is a rose.* ”
> **Gertrude Stein, 1874–1946, “Sacred Emily”**

R*obot is a robot is a robot is a robot.* Or is it? The very word spawns a multitude of mental images that vary with the person being queried. Some people think of “metallic men”: subservient beings that mirror humans but are driven by motors, pistons and circuits. Others see small and unassuming household helpers that resemble upturned buckets and keep floors clean or lawns trimmed, without constant supervision. Industrial types will picture hulking and bare mechanical arms that spot-weld a stream of car parts for days on end, never tiring and always with pinpoint accuracy. A number may even think of the red, amber and green collection of automated signals we usually call traffic lights. Some researchers have even taken the step of performing studies that investigate how robots are perceived by humans and the attitudes people take towards the idea of interacting with robots in their daily life [2, 26]. So what is a robot, and why does this one word bring up such different images?

1.1 Robots: Intelligent Machines or Glorified Toasters?

At their core, all these different *robots* have a point of commonality: they are human-built machines, designed to perform a task that humans would prefer not to do themselves, without the need for constant human supervision. The first point notes that robots are machines or tools, built by tool-users to assist with a task. The second point indicates that a robot performs a task that, in most circumstances, could otherwise be performed by a human. Why a machine and not a human is used varies: the task may be seen as too boring or repetitive, it may be more expensive to pay someone to do it, progress might not be fast enough if a person did it, it may be considered too dangerous for a person to do it, or it might be infeasible for a human to perform the task effectively with the technology of the day. The final point is what separates robots from most other machines. Robots do not need constant human supervision. They possess a means of monitoring their performance and making adjustments as required. The inclusion of performance monitoring and evaluation provides quality control, which in turn allows a greater operating precision than would otherwise be possible.

Even with these prerequisites for the prestigious title of “Robot”, the definition is grey.

In industrial terms, the ISO (International Organization for Standardization) definition of industrial robot is an "*automatically controlled, reprogrammable, multipurpose manipulator programmable in three or more axes*" [59]. On the home front, a humble washing machine with load-sensing ability apparently fits the bill. There can be no question that it is a machine built to perform a task that humans would rather not do and its various sensors and circuitry allow it to perform its task without supervision (although anyone who's ever put in too much detergent knows the occasional glance can't hurt!). It is clear, therefore, that the term robot is not very strictly defined and as a result encompasses a wide range of machines. The addition of one word to our machine's title will provide a more restrictive yet more useful term, that highlights how the systems of interest differ from a washing machine or industrial spot-welder. That word? *Intelligence*.

1.2 Intelligent Autonomy: Beyond Arms and Eyes

The concept of intelligence, and what makes something intelligent, is far too broad an area to cover in any detail here. The notion of artificial intelligence has been covered in great detail in many books, recent examples being Negnevistsky [100], Russell and Norvig [118] and Padhy [104]. With respect to robots, previous work has included investigating the components of an intelligent robot [28], what robot intelligence is [84], and what is expected of a supposedly intelligent robot [124]. Briefly we can look here at a number of applications of robots that are at least perceived to be intelligent before discussing in more detail the more commonly accepted notion of intelligence as it applies to modern robotics.

One of the best-known uses of robots in recent history has been in the realm of planetary exploration. Martian rovers like those described by Kemurdjian et al. [67], Muirhead [89] and Stone [131] have been used and will continue to be used to explore the surface of planets like Mars long before it is feasible for a human to set foot. Suburban disasters, most notably the collapse of buildings after events like earthquakes or terrorist attacks, have produced a number of USAR (Urban Search and Rescue) robots, such as those shown by Blitch [6], Mae et al. [79], Murphy [91] and Murphy et al. [93, 94, 95], whose task is to assist in the location and possibly retrieval of people trapped in unstable rubble. Military and law-enforcement organisations have long investigated the use of robots for both active tasks, like bomb disposal [146, 49] and patrolling [17, 18], and comparatively passive tasks, like mine detection [153], surveillance [29, 122] and scouting [145, 119, 120]. Other emergency operations have looked into the use of robots for fire-fighting [85] and handling hazardous materials such as chemical and radioactive spills [132].

In most of even these seemingly cutting-edge designs, robots are still not what you would call truly intelligent. Most of these robots are predominately teleoperated: operated by a human controller at a distance. They are usually a step above being completely remotely controlled (for example, they may be able to be told to drive ahead five metres, turn ninety degrees and drive another three metres), but the high-level decisions on what to do and when are still made by a human.

As useful as these types of robots are, their applications are limited. With their need for a human controller, their primary purpose is to separate humans from hazards, not reduce a human workload. Intelligent robotics aims to take these helpers a step further by giving them the ability to make their own decisions. The end goal is autonomy: robots that can think for themselves and act accordingly, robots that are more than just an extension of a human's

arms and eyes.

1.2.1 It's All About Balance

Having just extolled the virtues of robotic independence, it should be noted that there is much room for a useful mix between teleoperation and autonomous behaviour. Blending autonomy with teleoperation has been investigated in recent times, with respect to both general robotics [82, 50] and for specific applications, such as USAR [105] and space-based engineering [154]. In many situations it is perfectly acceptable, and perhaps preferable, to have a robot that can perform many of its tasks unassisted, but, for example, reports back to base and reverts to a teleoperated or decision assisted behaviour when it is unable to make a decision. In the same way that most human workers have some form of supervisor whose assistance is sought when necessary, so too can an intelligent and highly functional robot ask for help on occasion.

A good example of such a robot would be one whose task is to perform surveillance or area monitoring. It should ideally be able to navigate through its environment without assistance (perhaps after an initial training or familiarisation stage). It should probably also be able to discriminate between what is considered "normal" behaviour and that which is thought "suspicious". Given the enormous number of possible scenarios in which a person is behaving suspiciously – and the equally large number of potential responses – to expect a robot to be able to handle every situation is unreasonable. It is far more realistic to expect it to recognise a situation that it cannot handle and notify human supervisors, who can then decide what is best. In the end, the most useful and usable intelligent robots will undoubtedly be those that have the right blend of being able to make complex decisions autonomously and knowing when human assistance is required.

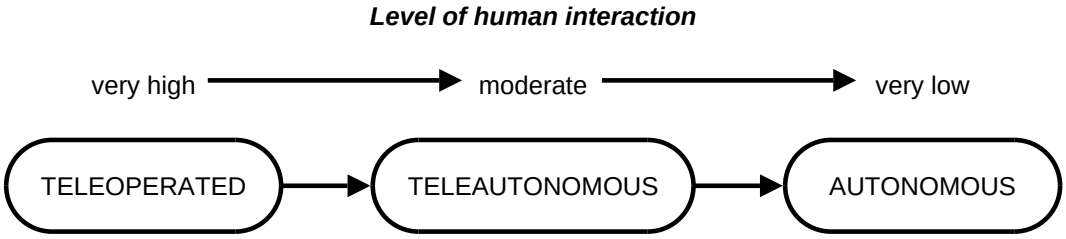


Figure 1.1: From teleoperation to full autonomy

Figure 1.1 depicts the transition from a teleoperated robot that requires a very high level of human control to an autonomous robot that has very low human supervision requirements. In the middle, we have teleautonomy, a blend of partial, remote, human control and independent robotic operation.

1.3 Requirements for Intelligent Robotics

We are now able to provide a succinct definition of the field: *Intelligent Robotics* is concerned primarily with *sensor informed, purposeful action*. An alternate definition is that "*an intelligent robot is a mechanical creature which can function autonomously*" [92]. The next logical question is "*what do we need in order to do this?*". Broadly speaking, an intelligent robot

will have three[1] required abilities:

SENSE: The ability to perceive the world,

REASON: The ability to process with intelligence what has been perceived, and

ACT: The ability to act upon the decisions made through processing what has been perceived, and hence modify the world.

The next section describes how these three core areas have traditionally been coupled together in an abstract sense. Following this is a more detailed discussion of the key areas – sensing, reasoning, control, actuation and dealing with uncertainty.

1.3.1 The Robot Paradigms

Throughout the development of robotics, the basic sense, reason and act abilities have been utilised and connected in various ways. Today, there are three basic design philosophies or paradigms that most roboticists follow: the hierarchical paradigm, the reactive paradigm, and the hybrid deliberative/reactive paradigm.

The Hierarchical Paradigm

Figure 1.2: The Hierarchical robot control paradigm

With an underlying goal in mind, the traditional hierarchical paradigm of intelligent robot control (Figure 1.2) sees the robot repeatedly sense, reason and act until the observed world is consistent with the end goal. Even a simple case of autonomously moving from one point to another consists of making an observation, determining how to move to become closer to the goal, and performing that movement. Here, the act of moving serves to modify the world by changing the robot's position in it. This approach provides a clean and logical flow of information that mimics how humans seem to approach even moderately complex tasks. Firstly, raw sensor data is obtained and processed, providing a higher-level description. (A human looks around, observing the location of obstacles and free space, for example.) Reasoning is then applied to this information to produce a plan at an even higher level of abstraction. (The human thinks about possible paths to their goal, evaluating each for overall suitability.) Finally, the plan is put into action when it is decomposed into realisable actions. (The human performs the motions required to traverse their chosen path.) The penalties, however, are typically high overheads: repeated reasoning operations are costly and will typically reduce overall performance in terms of speed and responsiveness.

[1]It should be noted that there is an at least partial movement towards the inclusion of a fourth basic ability: LEARN. To some extent, the ability to LEARN can be considered as being part of the REASONING process. This is still very much in its infancy, however, and is yet to be widely adopted [92]. As such, robotic learning will not be further considered here.

The Reactive Paradigm

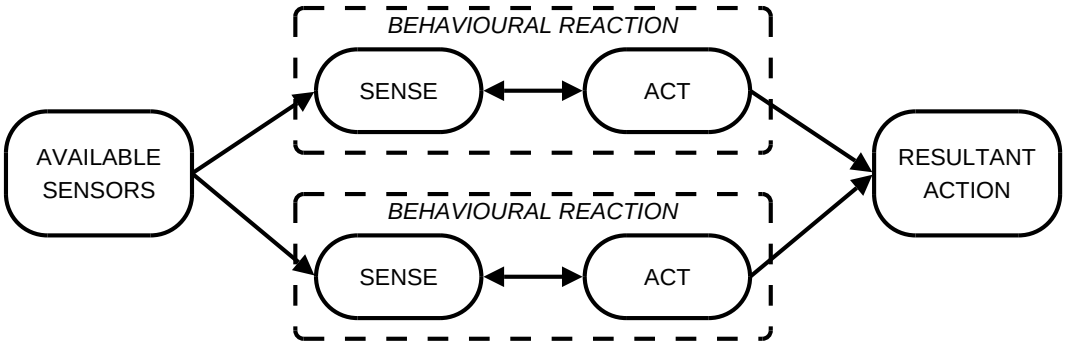


Figure 1.3: The Reactive robot control paradigm

An alternative approach essentially removes the reasoning stage, leaving a reactive system that acts based on short-term sensor input. Such an approach more closely copies the apparent behaviour of insects and other creatures of similarly low comparative intelligence. Most often, multiple actions are generated (perhaps from the input of different sensors or from different interpretations of the same sensor data) and the final action is a combination of the results. As shown in Figure 1.3, data from the available sensors is fed to multiple behavioural units, each of which produce an action. These actions are combined to produce a final, resultant robot action. For example, the action of driving straight ahead towards the observed goal can be combined with the collision-avoidance action of turning to the left to avoid an obstacle. The result may be to continue driving but at an angle, still heading towards to goal but avoiding the obstacle simultaneously. The reactive paradigm's greatest benefits are its simplicity and its high-speed operation.

The Hybrid Paradigm

Not surprisingly, current work follows a mixed approach. The Hybrid Deliberative/Reactive Paradigm [92] aims to take the best of both traditional approaches. At a high level, reasoning is performed in an intelligent manner that makes use of as much information as is necessary to produce a plan. Once formulated, the plan is decomposed into a series of subtasks that can then be realised through the execution of comparatively simple and largely reactive behaviours. Figure 1.4 shows how the behavioural, sense-act unit of the reactive paradigm is now treated as a primitive unit. Now, however, a deliberative reasoning unit oversees operation, receiving sensor input as required and controlling the selection of active behaviours as well as potentially having direct control over the resultant action. At various stages of operation the focus may shift between reasoning (deliberating) and responding (reacting). Often this will mean a change in the level of processing: some circumstances dictate a heavy emphasis on high-level planning and decision-making whilst others are better suited to a simpler reactive approach.

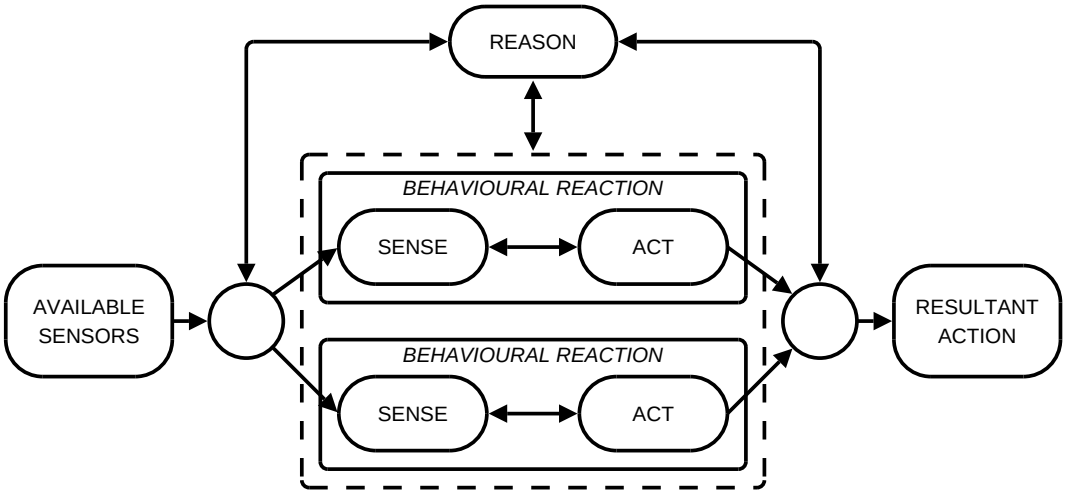


Figure 1.4: The Hybrid Deliberative/Reactive robot control paradigm

Conflict Resolution

While not explicitly shown in Figures 1.3 or 1.4, when utilising multiple, ultimately independent levels of reasoning and reactionary behaviours, a further task comes into play: conflict resolution. Depending on the specific design, it is entirely possible that resultant actions will be in conflict. In a simple example, a purely reactive robot may have two behaviours: move towards a light and don't bump into obstacles. With an obstacle placed directly between the robot and the attractive light-source (in a manner that won't block the light, of course), there is conflict between the desire to travel straight ahead towards the light and the desire to avoid the obstacle in front. One method of dealing with such conflict is to assign priorities to behaviours. A high-priority behaviour, such as a survival-based collision avoidance, would then override a low-priority behaviour, such as heading towards a light. An alternative approach – and indeed an entire design philosophy – is the subsumption architecture [11]. In this framework, low-level behaviours can be inhibited or subsumed by the next-higher level. Following the notion that a higher-level behaviour, or layer, is by definition of greater competence, conflicts are resolved simply by performing the operation suggested by the behaviour of greatest competence. This can also be viewed as a variation of priority-based resolution, whereby priority is related to competence and final control is absolute.

1.3.2 Sensing and Perceiving

Perception and sensing are what allow a robot to observe itself and its environment. They provide a means for gaining information about the position and configuration of various parts of the robot and for gathering information about the environment in which the robot is operating. If a robot cannot see, that is, it cannot sense and measure some characteristic of its environment, whether visually or otherwise, then it cannot detect objects, track objects, locate goals, find paths or control its motion. A blind robot is not much of a robot at all.

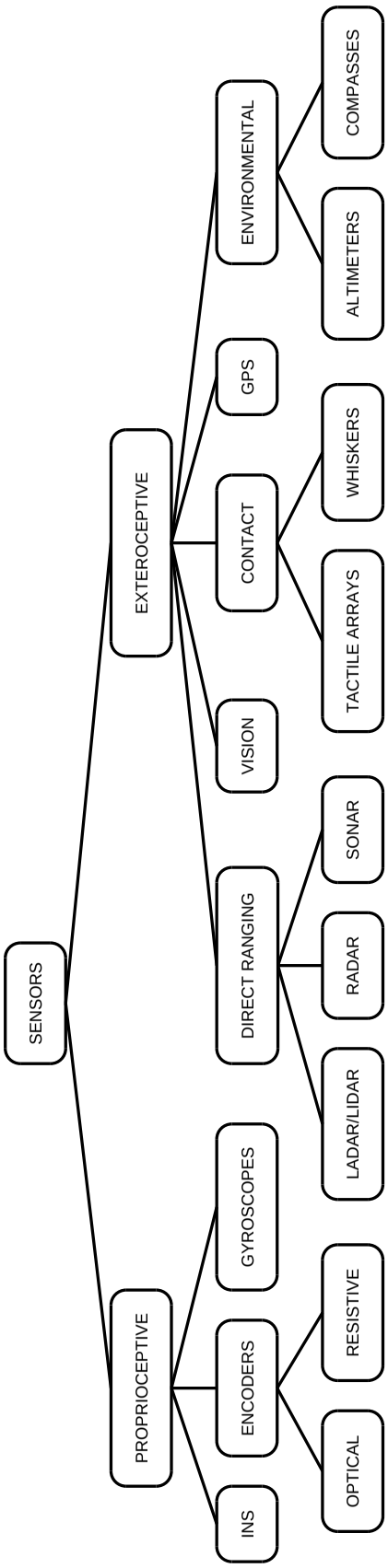


Figure 1.5: The relationship of robot sensors

A detailed survey of a great many sensors used in robotic applications is given by Everett [32] but a brief description will be provided here. Figure 1.5 gives one possible categorisation of many common robotic sensors. At a high level, sensors can be classed as either proprioceptive (measuring internal attributes) or exteroceptive (measuring external attributes). Proprioceptive sensors commonly include angular sensors to measure joint angles, shaft encoders to measure wheel rotations and INS (Inertial Navigation System) devices to measure a robot's overall angular and linear acceleration (and hence velocity and position, with appropriate initialisation). Such measurements allow a robot's movement (both overall and that of components like arms) to be monitored and controlled, without reference to the surrounding environment. This highlights a limitation of proprioceptive sensing: while able to measure the robot's actions, such as the rotation of a wheel, it is unable to measure the *effect* of those actions, such as how far the robot actually moved as a result of a wheel rotation.

Exteroceptive sensing, on the other hand, makes measurements of attributes external to the robot, that is, measurements of the environment. It should be noted that the distinction between exteroceptive and proprioceptive sensing is at times grey. For example, if a camera is used to observe and track a robot's own arm as it moves through a workspace, is that exteroceptive or proprioceptive? (The answer, in fact, is that this form of sensing would be considered exproprioceptive, that is, the measurement of self with respect to the external environment. The point, however, is that a single sensor can be used for different forms of sensing. In the remaining material, unless otherwise stated, we will be considering the *most common* usage of a sensor when classifying it.) As with internal sensing, the type of sensor used is highly dependent on the task being performed. Tactile sensors are used when contact is to be measured, whether in the form of simple bump-sensors or on a synthetic fingertip. When required, odour sensors can detect the intensity of an airborne odour and possibly discriminate between different odours. An example of their use is in the work done by Kowadlo [70] where naïve physics is combined with odour sensing to find the source of an odour. Ultrasound transducers allow the distances to surfaces to be measured with great accuracy, but limited range, largely due to the fairly rapid attenuation of sound waves in air. For example, Kleeman [68, 69] describes an advanced sonar sensor system capable of sensing and classifying planes, corners and edges at distances of up to five metres while moving at up to 0.6 metres per second.

In recent times, GPS (Global Positioning System) devices have revolutionised the process of localisation, whether of people, cars, tagged animals or robots. Using a constellation of orbiting satellites and high-precision timing devices, the location of objects on the Earth can be reliably determined to accuracies of ten to twenty metres [158]. The addition of signal correction through DGPS (Differential Global Positioning System) is able to increase accuracy to less than one metre [86]. The use of satellites is, however, one of the main sticking points of GPS: line of sight to multiple satellites (typically 4 or more) is required. In a number of environments, such as even moderately dense natural bushlands and forests, GPS usability is intermittent at best. As a result, GPS will be most useful and usable as a periodic localiser or for confirmation of location at places of known, reliable signal reception, for example in a clearing or at an intersection of tracks, where tree-cover is minimal.

In terms of their most common forms of usage, these types of sensing are all low-density. A single sensor reading typically consists of one or a small number of numerical values. Multiple sensors can be combined, for example, a matrix of tactile sensors covering a finger or palm or a sonar array, but the amount of information is still fairly small. For many tasks, such as navigation and in particular navigation outdoors, a greater information density is required to

allow ambiguity or uncertainty to be resolved, or simply to provide greater detail for greater precision.

A wide range of sensors, all using the properties of electromagnetic radiation, have been developed over the years that provide a far greater density of information in their common configurations. Two basic forms of sensor are widely available: vision sensors predominately use electromagnetic radiation in the visible spectrum (i.e. visible light) and range sensors use electromagnetic radiation of either greater wavelength than the visible spectrum (typically in the form of infrared LIDAR (Light Detection and Ranging)), or of shorter wavelength (such as RADAR (Radio Detection and Ranging) systems). Further details about these sensors and common usage are given in Section 2.1.1.

1.3.3 Intelligence and Reasoning

Once data has been collected, it needs to be processed and distilled. Some information requires very little processing to be used: an estimate of distance travelled can be found by simply multiplying the number of measured wheel rotations by the wheel diameter, for example. Other forms need multiple levels of pre-processing, processing and post-processing using complex algorithms to achieve the desired result: finding and recognising an individual face in an image is one such task.

One of the key elements in processing is data modelling. Rarely does raw data provide immediately usable information. More often, a model is used to represent data in a more meaningful and consistent manner. Given a task, a detailed description of the concepts it involves and a list of the objects and attributes required to be measured and monitored, a system model can be determined that clearly defines the key elements of the problem and the relationships between them. Models are, to some extent, an embodiment of the assumptions made about how the task can be carried out. For example and almost trivially, when navigating a wheeled platform it is usually assumed that the robot will be traversing on a predominately solid surface (that is, the ground) and that wheel circumference is constant. This leads to a model of motion consisting of circular wheels rolling over a typically piecewise-smooth surface, in turn leading to the simple relationship between the number of wheel rotations and distance travelled. Models transform raw sensor data into more abstract and problem-specific representations.

With the required information held in a meaningful representation, further processing can be done that attempts to apply notions of intelligence and reasoning to the problem. The aim is now to decide *what* must be done next. At this point a goal – sometimes a high-level goal describing the desired end result, sometimes an intermediate goal that serves as a stepping-stone – is taken into consideration and compared against. This allows an action to be formulated that will, hopefully, bring the robot closer to completing its task. Continuing with the navigation example, having determined its current location (perhaps by measuring the number of wheel rotations since initialisation), and knowing its desired destination, a robot can determine the movement required to reach the goal.

This step of planning future actions, deciding *how* to do what must be done, may very well be more complex than deciding *what* needs to be done. Typically in this process the thread of increasingly abstract data representations is unwound as high-level sub-goals are broken down into simpler and more primitive steps that are more directly realisable and able to be performed as actions.

Figure 1.6 graphically depicts this progression of data in a typical system. There is

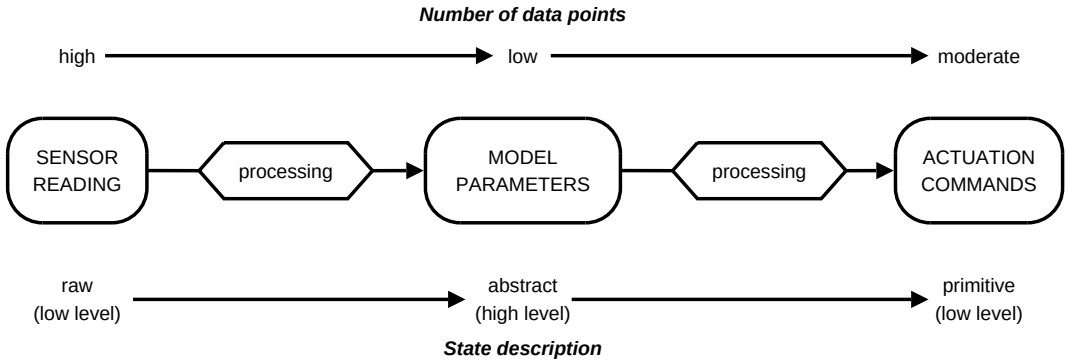


Figure 1.6: Typical levels of data modelling

usually a high number of data points available from a sensor, the data being in a low-level, raw form. For example, an image of thousands of pixels, each representing a light intensity at a point on the sensor. A predetermined model is then fitted to the data (perhaps after a few stages of processing), giving a comparitively low number of model parameters that provide a high-level, abstract description of the state. At this stage, the data has been transformed from a very general form to one that is highly task-specific. Continuing with the imaging example, this might mean finding the strongest pair of converging straight lines in the image (representing road edges, for example), perhaps after contrast adjustment and a thresholding operation, and describing these lines in terms of two simple linear equations, each with only two parameters (perhaps with estimated uncertainties). Further processing or interpretation of these parameters, taking into account goals and objectives, produces a moderately-sized set of actuation commands or signals that are once again low-level and somewhat primitive in nature. This could mean finding a centre-point between the discovered lines and determining a steering angle to use to point the robot directly at that point.

1.3.4 Control

Control systems, within the scope of robot navigation, appear at both the highest and lowest levels of abstraction. At the top end, deliberative control methods take in generally high-level descriptions of the environment and aim to intelligently process such data to plan actions (such as determining where to explore next). Reactive control systems sit at the lower end and, as the name suggests, typically react to immediate information with little regard for the overall state of operation.

Deliberative navigation control methods must take into account the notion of an underlying mission or goal. Typically, as much information as is available – both immediately and through continual accumulation – is used to determine a course of action. For example, exploratory traversal through an environment may be combined with the development of a map. This map will then in turn be used to determine which area should next be explored. In this manner, the underlying goal of exploration is combined with long term gathered knowledge to make an intelligent decision. Because of the large amount and potentially varied types of information processed, deliberative control systems are usually very complex and are often somewhat slow as a result.

At a lower level, reactive methods are used to respond to the immediate surrounds and allow for safe traversal. For example, having decided that it wishes to travel ahead ten metres, a robot can use relatively simple (in terms of traditional intelligence) reactive methods to respond to obstacles and conditions that it encounters along the ten-metre journey.

For some operations, reactive control may be all that is needed. A robot navigating based on a detailed set of provided waypoints could function with little more than the ability to work its way from point to point, dodging and manoeuvring around obstacles as they are encountered. As has been stressed many times previously though, the most useful robot will make use of a combination of high-level, intelligent deliberative navigation control as well as low-level, immediately responsive reactive control.

1.3.5 Actuation and Manipulation

The final stage in each sense-reason-act cycle takes the decisions that have been made and puts them into action. Having determined what we wish to do we must now actually do it. It is during this phase that we perform operations that will then change what we subsequently sense. In some scenarios this comes in the form of manipulation of objects. A household robot, for instance, may be trying to retrieve a cup of tea for its owner. At first this might mean simply moving an arm towards the cup, which changes the relative positioning of the cup and the arm. This change will be seen in the next sensing phase, resulting in a slightly modified perception of the current state of task completion. In turn, this produces a new action and the process repeats, until the hand is in a position to grasp the cup (or the cup and its contents are knocked over because the wrong action was chosen!).

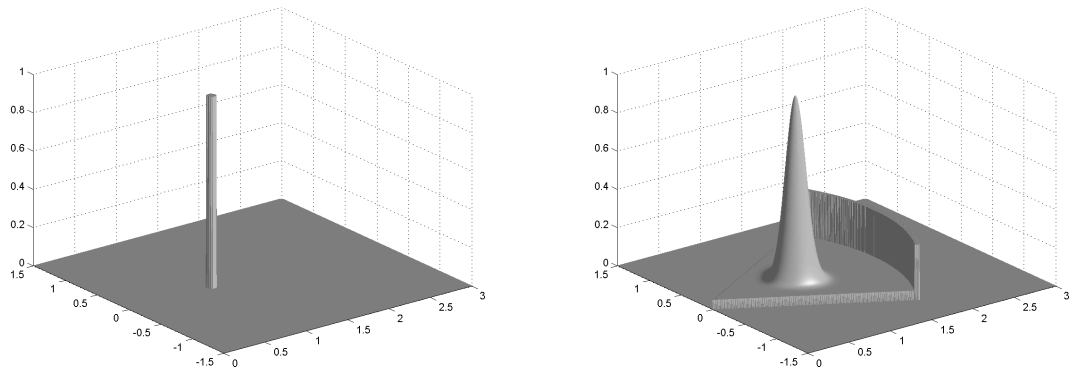
In another common scenario, a mobile robot undergoes movement, taking it a step – figuratively or literally – closer to its target location. Again, each incremental motion slightly alters the following perception of the world. Sometimes the changes will be slight and the modifications to action minor. At other times, action may cause a drastic change in the perceived environment. Travelling past a blind corner, for instance, can provide enough new information to require a significant change to the high-level plan. In either case, performance of an action provides the feedback necessary to close the gap between the current and desired state of operation.

1.3.6 Dealing with Uncertainty

It is perhaps useful to note the approach taken to intelligent handling of information by many if not most modern robotic systems. Errors and uncertainties appear in every facet of all non-trivial systems, robotic or otherwise. Even an answer the age-old question "*how long is a piece of string?*" involves uncertainty. If we use a standard ruler with millimetre markings, for example, our accuracy is bounded by the separation of markings; typically we would give a reading accurate to one half of the smallest measure (in this case half of a millimetre). A robotic system, measuring the speed of wheel rotation, the time-of-flight of a laser pulse or the contact force on a tactile sensor, is faced with similar bounds on accuracy. Many measurements involve the conversion of a continuous or very finely-resolved property (such as the intensity of light falling on an imaging sensor element or the resistance of a wire-wound potentiometer) into a coarser digital representation suitable for computer-based storage and manipulation. A conversion such as this introduces another source of uncertainty: quantisation error. Adding to these sources of informational fuzziness, the models we use to

represent important features are themselves approximations of some underlying principle. Even binary decisions ("*is there an obstacle in front of us?*") are most often determined by thresholding some otherwise fuzzy piece of data.

To accommodate the many sources of error and uncertainty that cannot always be ignored, modern robotic systems tend to take a statistical or probabilistic approach to most problems (for example see [141]). Changing binary decisions, for example, to a numerical degree of likelihood ("*there is a ninety percent chance that there is an obstacle in front of us*") can provide two key benefits: uncertainties can be propagated through the decision-making process (meaning the final decision is potentially better informed because as much initial information is retained as possible), and more meaningful comparisons of values can be made (traditional logic does not provide for "*degrees of truth or falsehood*").



(a) Idealised sensor model

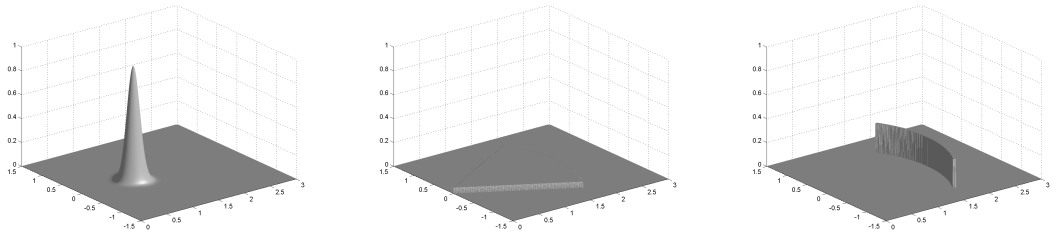
(b) Probabilistic sensor model

Figure 1.7: Ideal and probabilistic sensor beam models

Figure 1.7 shows the difference between an idealised model of a beam-based sensor (such as a sonar sensor) and a more realistic probabilistic model. (The model shown here is adapted from a common beam model given in [141] and is also similar to that used in [92].) In this hypothetical example, an object that is essentially a point exists a short distance away and directly in front of the sensor, which is located at the origin. In Figures 1.7 and 1.8 the horizontal axes represent distances from the sensor while the vertical axis represents a measure of likelihood of obtaining a reading at the corresponding position, normalised between zero (impossible) and one (highly likely).

In the idealised model (Figure 1.7(a)), only one possibility exists: a single reading at the correct object location. Everywhere else, the model gives zero likelihood, indicating that the sensor will never give a reading that is not the true location of the object. The probabilistic model, however (Figure 1.7(b)), is quite different. Outside a wedge-shaped region defining the limits of the sensor beam, the model also gives zero likelihood. Within the wedge, the model accounts for a number of possibilities.

Figure 1.8(a) shows the component that describes a correct reading with local measurement noise (the probability of a hit, or p_{hit}), modelled with a narrow Gaussian distribution centred on the correct position. The component in Figure 1.8(b) accounts for unexplainable random measurements (the probability of a random measurement, or p_{rand}). Such readings



(a) p_{hit} (b) p_{rand} (c) p_{max}

Figure 1.8: Probabilistic sensor beam model components

can come from a variety of places, such as crosstalk between multiple sensors, electronic interference and phantom sonar echos. These errors are modelled by a uniform distribution within the beam wedge. The final component in Figure 1.8(c) corresponds to sensor failures (the probability of a maximum reading, or p_{max}), those being occasions where the sensor fails to detect the object and instead returns some maximum measurement. This is modelled as a line-mass distribution along the maximum-range edge of the beam wedge. When combined (as in Figure 1.7(b)), the model describes a sensor that is most likely to give a reading close to the correct measurement, but may also sometimes give a false maximum reading and less often a completely random erroneous measurement.

The central notion is that errors and uncertainties exist and, while improvements in sensors, mechanics and computers can reduce many of these, the most useful approach is one that accepts such uncertainty and deals with it gracefully. Statistical and probabilistic handling of information is one approach that can make use of a wealth of well-established background knowledge. Mathematical theories of probability, for example, have been around since the 17th century [100].

1.4 Navigation: Getting from A to B

Navigation is the process of determining how a target destination can be reached from a starting position (often the current position). This can be done at a global level, where the absolute position of the starting-point, destination, obstacles, paths and other relevant features, are known and used. It can also be done in a reactive sense, where little more than what can be immediately sensed is used. As with many things, the most useful and effective implementation will typically see a balance between these extremes. In a great many environments, some knowledge exists and can be used prior to entry: buildings have floor-plans, urban roads are mapped out and even parks and forests have many major trails and tracks roughly mapped. Where such information is available, it is highly advantageous to make use of it to assist navigational decision-making. Even so, things can and do change: furniture can be moved and people rarely stand still for long, vehicles travel at high-speed down roads and the roads themselves can sometimes be closed, and weather, time and nature can cause bush tracks to meander or even disappear. Clearly, the ability to sense and adapt to the current state, however much that may deviate from what was expected, is an important ability.

A more detailed background review of robotic navigation is given in Chapter 2, including descriptions of navigation components, the classes of environment in which navigation must be performed and some details of specific pieces of relevant work on robotic navigation. In the following we briefly cover some of the more immediately relevant areas.

1.4.1 Bush Environments

With a great many people living and working in fairly densely populated areas, whether they be bustling cities or more subdued suburbs, it is not surprising that much of the work done in the area of robotic navigation, and navigation for service robots in particular, has focused on navigating in urban environments. This includes indoor home and office areas as well as outdoors around buildings or along motorways.

This has left many less-populated but no less important areas comparatively unexplored. One such environment is that of the bush and forest. In eastern and south-eastern Australia, for example, there are huge expanses of dense bush and scrub, much of which is contained in State and National Parks. Why, though, is this area important to robotic navigation research? There are a number of groups that work in these areas, all of whom could benefit from service robots. Bush firefighters are often called upon to risk their lives to control the many bushfires that break out in the warmer months. Search-and-Rescue crews are often required to locate, assist and extract lost or injured bush-walkers at all times of the year. Park maintenance personnel have many hundreds of kilometres of tracks to patrol and maintain, looking for fallen trees to clear or washed-out tracks to be repaired or re-routed. In all cases, service robots that can self-navigate through these areas and report when anomalies are found would free valuable human resources that could be more efficiently utilised.

1.4.2 Vision for Navigation

Humans have millennia of experience using their own vision for many tasks, navigation included. Many of the environments we work in have been modified to make our own visual navigation easier: signs on doors and buildings, painted arrows telling us where to go and lane markings on roads are just some of these.

There are, of course, alternatives to vision for robot navigation sensing: GPS is perhaps the most common localisation sensor, while sonar and LIDAR are widely used, as ranging sensors, for everything from obstacle detection to terrain modelling to localisation.

These all have limitations and problems, especially in bush environments. GPS, while undoubtedly a revolution for localisation all over the globe, can be highly unreliable under the cover of even moderately dense tree canopies. The result is that it cannot be relied upon for continual use. In addition GPS is a localisation mechanism only. While it can say with great accuracy where you are, it cannot say whether the track ahead is clear or impassible. In the very least, additional sensors are necessary to provide dynamic information about the immediate environment. In bush environments, GPS is best used as a secondary sensor for intermittent confirmation of position, for example to periodically nullify errors from odometric drift. Sonar has limited range, and can be highly sensitive to environmental conditions, such as air temperature, which can easily change. It is also not able to provide a dense enough set of readings in an environment where there is a high degree of natural irregularity and variability. LIDAR is a highly popular choice for outdoor range sensing. With distances of anything from tens of centimetres to hundreds of metres able to be measured, range is not an issue. Because

LIDAR scanners typically provide only a planar scan, however, overall information density is fairly low. While a planar scan is appropriate when traversing flat ground (for example, indoors or outside on flat, paved roads or pathways), even moderately rough terrain can necessitate a greater amount of data. Pan-tilt units can be employed to provide a volume scan, although these can be slow or complicated to control when compensating for the unit's inertia and resultant mechanical vibration. Such mechanisms introduce an additional complexity that reduces the appeal of the LIDAR's initial simplicity.

Because of the rich and dense information available, vision has long been a popular choice for tackling problems of navigation. There are many aspects of the environment, colour and texture for example, that are either impossible or very difficult to measure with other sensors, yet have the potential to provide a wealth of usable information. There are many ways in which vision can be used, from stereo-based ranging to colour analysis to motion estimation, that make it suitable and a good choice for outdoor navigation.

Another benefit of typical vision systems is cost. At the cheapest end of the scale, simple USB (Universal Serial Bus)-based web-cams can be purchased for under one-hundred Australian dollars. Even a more complicated, albeit more flexible solution consisting of a frame-grabber and camera can be obtained for under three-hundred Australian dollars. A typical LIDAR system, on the other hand, can easily cost in the order of multiple thousands of dollars. Additionally, vision system parts (or replacements if required) are commonly available consumer products, meaning they can be easily purchased from a greater number and variety of suppliers than sonar or laser-based products, whose typical use is in industrial or commercial settings.

1.5 Visual Navigation on Rough Tracks in Bush Environments

As mentioned, the bush environment is an area that has largely been overlooked by robotic researchers, yet there are a number of working groups, many of whom risk their lives for the benefit of others, that could greatly be assisted by service robots. With the most useful robots being those that have an appropriate blend of autonomy and human control, robots that can self-navigate in these environments and report back when faced with a problem have the potential to be both useful and viable.

For example, a firefighting robot could be deployed near where a fire is being actively fought and then make its way along some fire-trails to a useful vantage point. From here it could send back real-time data about temperature, humidity, wind speed and direction, and many other pieces of information that could help firefighters plan their attack or their retreat if conditions became threatening. All this could be done without taking people away from the important task of actually fighting the fire. Search-and-rescue crews could be assisted in their task of searching along the many hundreds of tracks that exist in State and National Parks for lost or injured bush-walkers. If equipped with thermal cameras, for example, they could look for body warmth and report back to base when something is found, allowing a much larger area to be covered in less time. Park rangers, who have potentially thousands of kilometres of tracks to inspect and keep maintained (possibly to ensure clear access to fire crews in an emergency) could be greatly helped by robots. A robot that can systematically work its way through the many tracks in a park could be used to notify rangers when track blockages are found. In all cases, the task of the human worker is made easier and potentially safer by the use of what is ultimately an expendable machine.

A common element in all these scenarios is the ability of a robot to navigate autonomously, and, in particular, to navigate along the many rough and irregular tracks that work their way through the parks and bushlands. While cross-country traversal is an option (and at times a necessity), roads provide a number of distinct advantages:

Continuity For the most part (with the exception of occasional and unforeseen blockages) roads are continuously traversable. If there is a road from point A to point B it means, by definition, that there exists a path between those points that has a high likelihood (factoring in environmental changes and degradation) of being navigable and traversable.

Smoothness Roads and tracks are, in terms of changes in direction, smooth passageways that typically only deviate from a relatively straight line or gentle curve when the environment dictates, for example, to go around a large rock formation or if a straight-line path would produce too steep a path.

Prior intelligence Roads are created to provide easy access to important places. As such, they represent passageways between locations that were deemed to be important and therefore are likely to be important places to visit again. They have also, at some level, been planned and so represent passageways that are in some way *efficient*. If a significantly better path exists, it is highly likely that at some point a new road would have been created to take advantage of that path. A common example of this can be seen in many open spaces that have high foot-traffic, such as university campuses. Very often, while paths have been planned and concrete pavements laid we see well-worn paths across lawns that clearly indicate where people *actually* travel, having decided the concrete is not the best option. Traversal along a road implicitly takes a path that has already been planned.

In determining how navigation in the bush can be performed and, specifically, determining an appropriate sensor modality, a few key factors can be considered. Firstly, when considering the low budgets the relevant organisations are most often faced with, the cost of the overall robot is of great importance: an unaffordable tool is of no use to anyone. The fact that these robots could be operating in potentially dangerous situations, where they could easily be damaged or completely destroyed, only serves to increase the need for a cost-effective solution. Especially when considering the range of ways data can be interpreted and used, vision represents excellent value. Secondly, the context of use strongly suggests that the most effective deployment of this type of service robot would see multiple units being used simultaneously. Even if not working collaboratively (and a collaborative approach is beyond the scope of the work presented here), there is every chance that two deployed robots would at some stage encounter each-other. Passive vision greatly simplifies such encounters by avoiding altogether issues associated with sensor interference (such as laser or sonar cross-talk). Thirdly, as eluded to earlier, the most useful robot in these operations will be one that blends autonomy with teleoperation, allowing a human operator – most likely one with minimal training – to take over if and when necessary. As such, it is very important that sensor information can be presented in an intuitively understandable and usable representation. While LIDAR or sonar information can be processed and presented in a form amenable to human interpretation, vision information is typically far richer, requires no processing to be usable and is very natural and intuitive. Finally, the extensive range of computer-vision and image-processing research that has already been performed means there is a great wealth

of knowledge, methods and techniques that can potentially be used to perform many tasks required for robot navigation. It can be seen that passive vision is a highly appropriate choice for the task at hand. We therefore arrive at the proposed area of research: *Visual Navigation on Rough Tracks in Bush Environments*.

1.6 Contributions of this Thesis

This thesis presents several coherent pieces of work that together form a navigation system for an outdoor mobile robot, operating in poorly structured natural environments, typified by State and National Parks in South-Eastern Australia. Specifically, the focus is on navigation along rough tracks and dirt roads.

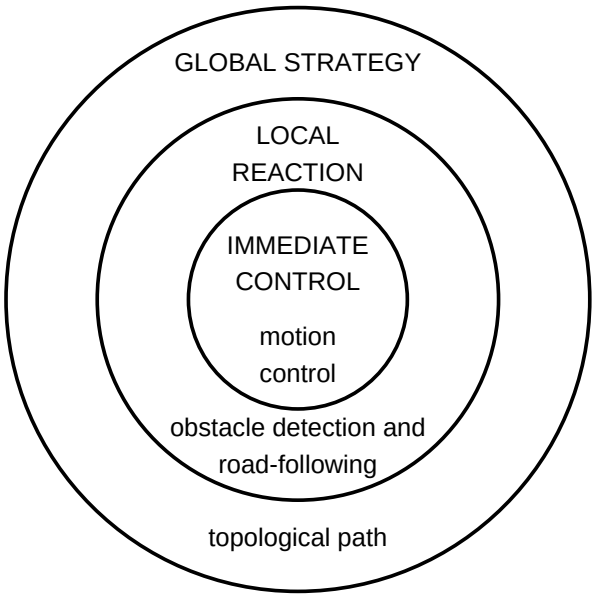


Figure 1.9: Levels of planning

Figure 1.9 gives a graphical representation of what is being done and how the components are linked.

At the innermost level, *Immediate Control*, we aim to provide a mechanism that is capable of performing relative localisation, primarily for the purpose of motion control when traversing over rough and irregular terrain, where traction is often quite poor. In such cases, wheel slippage is highly likely, rendering traditional encoder-based solutions unusable. In a novel approach, optic flow techniques are used to visually estimate the motion of a robot over a variety of outdoor ground surfaces and provide odometry information that is independent of how the motion occurred (intentional or accidental).

A step outwards, at the level of *Local Reaction*, we aim to provide a combination of subsystems that can sense the immediate environment and find a suitable path, based on a higher level goal. In the environment being investigated, this means two things: detecting potential obstacles and following poorly structured dirt tracks. For the purpose of obstacle detection, a stereo vision system is used that can provide a dense two-and-a-half dimensional view of the world in front of the robot. Obstacles are any entities, whether distinctly separate

from the otherwise fixed environment or not, that would in some way restrict or hinder movement or be potential unsafe to attempt to cross. In the target environment, this will most often be surface irregularities – for example ruts, potholes or ridges – or debris-like material such as fallen branches, large rocks or exposed tree-roots. Using a depth-map and a locally estimated ground plane created from stereo-vision, a likelihood map is generated that describes the likelihood of an image region being an obstacle. To take advantage of the many tracks that exist, a system capable of detecting and describing poorly-structured dirt roads using colour vision is used. This subsystem uses statistical techniques to firstly describe the road surface in terms of colour-space statistics and then classify colour images into regions that appear like the road and those that do not. The end result is a likelihood map, an image that describes where the road is most likely to be. These two likelihood maps are then combined to determine an overall *"amenability map"*, a map that highlights terrain that is both likely to be the road and unlikely to contain an obstacle. With the navigational goal here being to drive along the road while avoiding obstacles, a set of potential paths can be generated and evaluated to determine a local path that best fits these goals.

Finally, at the broadest level, *Global Strategy*, we aim to take advantage of what is known about these environments in order to perform high-level path planning. These environments have most often been mapped in terms of the tracks that cover them and this information should be used as much as possible. Accordingly, a path planner using a hybrid metric-topological map is developed that represents the system of roads and tracks as an abstract graph, where edges correspond to roads and tracks (with approximately known lengths) and where nodes correspond to *"interesting locations"* (typically intersections and semi-regularly spaced points along tracks). While it is assumed that the locations of nodes are known (for example, they would be chosen as locations where there is clear GPS reception, or are suitable for some other localisation mechanism) and the approximate length of tracks between nodes is known, no other assumptions of geometry are used. The shape and structure of the intermediate tracks, for example, are assumed unknown. For any given pair of connected nodes, we simply know that we have two locations that can be detected with some reliability and that these two locations are connected by a track of an approximately known length.

1.7 Document Organisation

This chapter described broadly the general field of robotics, some of its origins, where it may be heading, and brought the work being presented into context. The remaining chapters are described as follows:

Chapter 2, A Background in Robotic Navigation, gives a general review of relevant areas of robot navigation. Firstly discussed are the main components that together form a complete navigation system, followed by an overview of the classes of environment in which robots operate. Details of the state of the art in outdoor robot navigation is then provided, with an emphasis on operation in unstructured terrestrial environments. In particular, work done in the areas of road detection and following, and improvements to odometry estimation are focused upon.

Chapter 3, Relative Localisation with Visual Odometry, describes a novel approach to mobile robot odometry that uses optic flow techniques to estimate the rotation and two-dimensional translation of a platform over a variety of surface types. The use

of exteroceptive sensing for odometry measurement ensures that all motions, whether intended or not, are accounted for. This is particularly important in an outdoor environment where the ground can commonly be quite slippery.

Chapter 4, Obstacle Detection with Passive Stereo Vision, presents a passive stereo vision system that provides the robot with the ability to detect and hence avoid obstacles. An obstacle detection mechanism can be considered to be one of the fundamental requirements of a navigation system for a mobile robot. Stereo vision can provide a dense two-and-a-half dimensional environmental map that allows the detection of positive obstacles (those above the ground surface such as rocks or fallen branches), negative obstacles (those below the ground surface such as potholes or trenches) and overhanging obstacles (such as low branches).

Chapter 5, Detecting Poorly Structured Dirt Roads using Colour Vision, provides details of a vision processing system that can detect poorly structured dirt roads. The use of statistical road characterisation and simple geometric modelling allow operation in a wide range of environmental conditions and at high levels of road degradation. This element provides a means for reactive path planning that takes advantage of what little structure exists in many bush environments.

Chapter 6, Local Path-Planning using Likelihood Maps, details a reactive, local path-planner is that uses the likelihood maps created by the systems detailed in Chapters 4 and 5 to determine a safe trajectory that stays on the road but avoids obstacles, taking into account further factors such as path length, a desired relative bearing and perceived path safety.

Chapter 7, Global Path-Planning with a Hybrid Map, describes an approach to high-level path-planning that is particularly applicable to navigation in bush environments that are approximately mapped. A hybrid map combining metric description of discrete locations (nodes, typically associated with a junction or an important point along a road) with topological connectivity via roads (edges) is used as a basis for global path-planning. The inclusion of "*pseudo-nodes*", map nodes that do not correspond to particular geographic features, provides an opportunity to exploit the benefits of cross-country travel when appropriate.

Chapter 8, Conclusions and Future Work, summarises the findings presented in each body chapter and gives an overall summary and conclusion of the research presented in this thesis, with reference to the goals stated in this chapter. In addition, a short discussion is given on how the current work could be improved and extended through future work.

Appendices: This document concludes with three appendices. The first presents a brief overview of the mathematics of stereo vision with respect to its use in Chapter 4. The second gives a list of publications that have arisen from the research presented in this thesis. The final appendix describes additional, video-based results included on the accompanying CD-ROM which augment the results provided in Chapters 4, 5 and 6.

Chapter 2

A Background in Robotic Navigation

“ Navigation was always a difficult art,
Though with only one ship and one bell:
And he feared he must really decline, for his part,
Undertaking another as well. ”
Lewis Carroll, 1832–1898, “The Hunting of the Snark”

As soon as a robot becomes mobile, navigation becomes relevant and indeed highly important. A robot that cannot make its way through the environment effectively and efficiently is of limited use, although some robots, such as those that randomly wander and vacuum floors, make do with no real navigation, or in the very least, with no *intelligent* navigation. A complete and exhaustive survey of robotic navigation is beyond the scope of this thesis. In presenting an overview of the topic, it is useful to divide the material into two broad areas: the central components that make up a navigation system, and the environments and robots in and with which these components will be used. This chapter concludes with a detailed look at some of the more directly relevant pieces of work in terms of the methods and techniques being employed.

2.1 Navigation Components

Comprehensive reports of the history and state of outdoor and general robot navigation are given by Spero [127] and Borenstein et al. [8] respectively. The general area of robot navigation can be subdivided into four narrower areas: sensing, mapping, localisation and path planning. While each of these fields can provide complex and highly capable solutions to navigation problems, more often than not, all will be required in some form if a robot is to move autonomously and intelligently through its environment.

Figure 2.1 shows the four areas of navigation mentioned above with the addition of two further areas that commonly exist in modern robotic navigation systems: SLAM (Simultaneous Localisation and Mapping) and landmarking. Each of these six areas are discussed

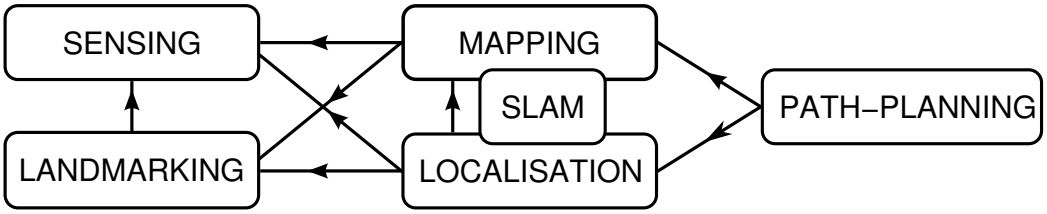


Figure 2.1: Navigation components

in greater detail in the following sections. In Figure 2.1, arrows indicate a dependency or requirement of a component upon another component. Path-planning needs a map and the ability of the robot to localise. Localisation requires a map, sensing ability and the identification of landmarks (if a landmark-based methodology is being used). Mapping requires sensing and (again optionally) the detection and identification of landmarks. Landmarking requires the ability to sense, in order to detect and identify landmarks. SLAM, being the process of localising *and* mapping simultaneously, is depicted as an overlapping of localisation and mapping and hence has the same requirements as these two components.

The following sections give further details of these six areas of navigation, with an emphasis on their application to navigation in largely natural outdoor environments.

2.1.1 Sensing

A thorough survey of many robotic sensors is given by Everett [32], with an emphasis on commercially obtainable end-products. Sensing, in the context of navigation, involves the acquisition of environmental information that can tell a robot where environmental elements, both static (such as trees and roads) and dynamic (such as people or other robots) are situated. Equipment for performing such duties has typically been one or a combination of RADAR, LIDAR, vision, ultrasonic ranging and tactile sensing, in approximate order of decreasing usable range. Radio or light ranging and vision sensing methods are very well suited to outdoor environments, in part because of their great range and usability in environments with rough and highly irregular surfaces. Correspondingly, a more detailed description of their use is presented here. Figure 2.2 shows one possible categorisation of many of these sensing mechanisms, divided firstly into ranging and vision methods, both of which are detailed below.

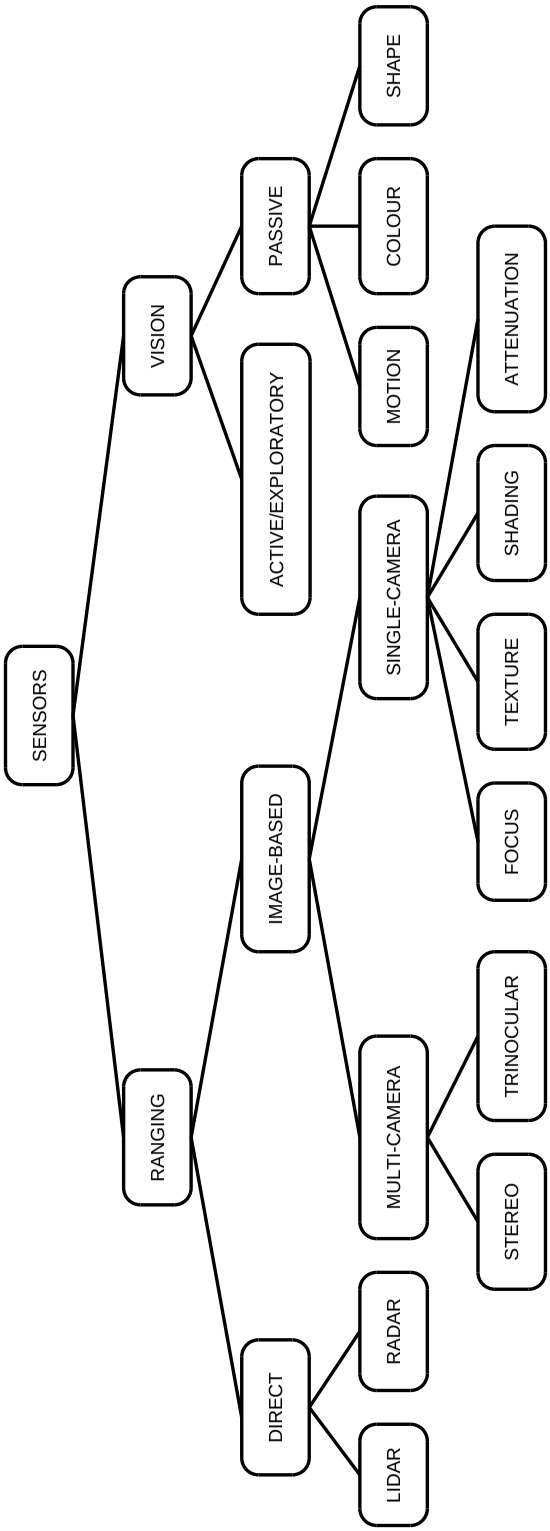


Figure 2.2: Sensors using light or radio waves.

Ranging

Ranging is the process of determining the distance (and usually the bearing or other form of direction) to an object, whether it be a fixed part of the environment or an obstacle or other similarly dynamic feature. Following Figure 2.2, a subdivision can be made based on how the range data is obtained, that is, either directly or through images.

Direct Ranging: A range of works covering the use of lasers for distance measurement is presented by Bosch and Lescure [9]. RADAR and LIDAR have the distinct benefit of providing two-and-a-half-dimensional data directly: a reading provides the distance to a reflective surface, so, knowing the direction of the emitted beam, the result is a three-dimensional vector from the sensor to the surface. (The limitation of a single viewpoint gives rise to the two-and-a-half dimensions of overall representation.)

This also highlights a potential problem. RADAR and LIDAR are both forms of energy-emissive sensing. They gather information about the environment by emitting controlled energy and measuring the effect of the environment on the emitted energy, such measuring the time-of-flight, phase-shift or frequency-modulation (with Doppler-shift being a common method) of reflected beams. Details of the mathematics and physical processes behind these methods is presented by Everett [32]. The energy-emitting nature of these modalities makes them particularly susceptible to interference from other sources emitting similar forms of energy. This may come from other sensors in the vicinity (such as from other robots) or from far more uncontrollable sources, such as our very own Sun. Great care must be taken to ensure that firstly interference is minimised (unique laser pulse shapes and matched filtering techniques can ensure only one's own signals are heard) and secondly that interference is handled both gracefully and safely. A second problem with emissive sensing, in certain operational conditions, is the simple fact that, if a robot can measure the energy it has emitted, so can somebody else. For a great number of operations this is of no consequence, but for missions involving secrecy, covertness or discretion, such as surveillance and military scenarios, being seen can be disastrous.

Image-based Ranging: Vision sensors, essentially just cameras of some form, can also be used to perform ranging. One form uses a structured light pattern, often a stripe or grid, which is projected onto the workspace and viewed by a standard vision sensor. By measuring properties such as the distortion of the known pattern, much can be deduced about the environment, such as the shape and position of objects and surfaces. This method also suffers from many of the problems of sonar, LIDAR and RADAR range-finders, most notably interference and detectability.

A second and highly popular method, especially with the increase in computing power available, is stereo vision. The act of projecting a point in three-dimensions down onto a two-dimensional image plane means image features can only be directly resolved in two dimensions (with distance from the camera unknown) when a single image is used. Stereo vision makes use of two images (or more, in the case of trinocular and higher order systems) taken simultaneously and from different positions. This allows the third dimension of image points to be resolved, as long as the point is visible in both images. Arguably, the single biggest challenge in stereo vision processing is the correspondence problem: determining where a known point (or other feature) in one image can be found in other images. A

thorough evaluation of many of the currently leading methods is provided by Scharstein and Szeliski [123].

There exist also a number of other methods that use images in combination with other known parameters to determine range. Such techniques include shape-from-focus, defocus or blur (using the effects of varied focal settings), shape-from-texture (using the distortion of surface texture), shape-from-shading (using the effects of surface shape on light reflection) and depth-from-attenuation (using the effects of light attenuation through a medium). A number of vision-based ranging techniques are presented by Jarvis [61].

Vision Sensing

While vision sensors can be used to determine range, as outlined above, this is but a small subset of the possibilities of vision. A great number of problems have potential solutions in the use of vision, from face recognition to motion analysis. Some such uses are presented here, divided into two sub-categories: active vision and passive vision. Here, active vision refers to systems that have direct control over the position or orientation (or both) of the vision sensor, that is, the sensor can be actively moved as part of the sensing operation. Compare this to a common alternative definition of active vision that refers to the active emission of energy as part of the sensing operation, for example, the projection of light stripes into the workspace. Passive vision systems, on the other hand, either cannot move the sensor (i.e. it is rigidly attached), or simply do not make use of the ability to move the sensor as part of processing or image acquisition.

Active Vision: Active (or exploratory) vision serves to make a vision sensor a movable part of the robot and finds use in a wide variety of scenarios. One instance of this sees the sensor mounted on a simple pan-tilt unit. This can allow a robot to track a moving target over a much wider area than would otherwise be possible. A biological analogy is the ability of most animals to move their eyes and head around and hence change where they are looking.

Another form might have a sensor at the end of a highly manoeuvrable arm. This can allow an object to be viewed from a great many different viewpoints, allowing it to be more accurately described and hence identified. This can be particularly important when attempting the manipulation of delicate objects in a relatively unconstrained environment (such as a home). A detailed inspection can provide the information required to form a suitable grasping motion. Environmental constraints can also make exploratory vision highly desirable if not a necessity. In many USAR scenarios, space is highly limited, and the ability to see through narrow gaps or around tight corners can mean the difference between saving a victim and passing them by. Making the sensor independently manoeuvrable – as opposed to only being able to change the point of view by moving the entire robot – can also potentially make planning easier, as there is much less likelihood of the robot making contact with an obstacle if only a small part of it is moving while performing a visual scan.

Passive Vision: The final sensing method covered here is also quite possibly the most popular. Humans are visually oriented beings and it makes sense that we try to make use of all we have learnt through visual observation. A significant portion of passive vision work can be classed as the detection and possibly description and recognition of objects. The type of object detected varies widely and can include simple geometric components (such as lines and circles), written characters, household objects, the horizon, stars, roads, human faces

and bodies, physiological features (such as joint cartilage or abnormal growths) and hand-gestures. All of these tasks utilise models of the object of interest, expressed in terms of image features (in simplest terms pixel values and locations). The creation of an appropriate model – one which both describes the object with enough detail to resolve potential ambiguity but is simple enough to be both computationally manageable and robust against expected variation – is perhaps one of the greatest challenges of vision processing.

Another large area of passive vision processing centres on motion. Motion analysis can allow moving targets to be detected and tracked for surveillance, unknown objects in a scene to be segmented and separated, and vehicles to be controlled. Because of the critical dependence on timing of motion analysis, it is only with the increase in computing power in recent times that many methods of estimating and analysing motion have become viable for use.

2.1.2 Landmarks

Because of the use of landmarks for both mapping and localisation, processes which themselves are very closely tied together, the topic of navigational landmarks will be treated separately here. A landmark is, quite simply, a feature of the environment that can be used as a point of positional reference, whether in absolute (global) or relative (local) terms. It could be something obvious like a sign, it could be something more subtle like an open clearing. Things like building entrances, doorways, staircases, signs, buildings, trees and large boulders can all be used as landmarks.

Landmarks also come in two flavours, commonly referred to as natural and artificial. These terms can be confusing, however, as in this context natural and artificial do not refer to organic and man-made [92]. A natural landmark is simply one that exists in the environment without regard for the task of navigation. That is, it has not been placed to aid navigation. As such, trees, boulders, buildings and bridges are all considered to be natural landmarks. An artificial landmark therefore is one that *has* been placed with the specific purpose of aiding navigation. Signs, flags, beacons and other such markers are all artificial landmarks.

While anything that can be sensed is potentially a landmark, for something to be a *good* landmark, it must posses certain qualities:

Positional predictability: The position of a good landmark must be predictable. Once a landmark has been seen and noted for the first time, future viewing must be able to provide reliable positioning data. In the significant majority of cases this will simply mean that a good landmark is stationary, although if enough information is known about a moving landmark (such as knowing its trajectory and starting position and time), it could, in theory, be used, albeit at the cost of much greater complexity.

Distinction: A good landmark is distinct, both in terms of being able to be separated from its surrounds and in terms of being able to be identified. Ideally, a landmark is a globally unique feature of the environment. Uluru in Central Australia, for example, is a good landmark because there is only one Uluru and it clearly stands out. If, for example, you measure your position to be one-hundred metres south of Uluru, there is no ambiguity. Things like trees, on the other hand, often stand out from the surrounds (unless you happen to be in the middle of a pine plantation, for example), but they all appear very similar. Further work then needs to be done to create a unique feature. For example, while an individual tree

may not be able to be uniquely identified, a particular set of trees with a distinct geometric layout may be at least unique within some local region.

Visibility: A good landmark is visible from a wide range of positions. Quite simply, the more often a landmark can be seen, the more often it can be used. Being visible over a large area also assists with continuity: a widely visible and distinct landmark (or set of landmarks) makes it easier to deduce spatial relationships between other landmarks that are mutually exclusive in terms of visibility. For example, in circumnavigating a highly visible central landmark from a distance, there will likely be many other landmarks that come in and out of view. If the central landmark remains visible throughout, it provides a stable point of reference by which the relative locations of the transient landmarks can be determined.

2.1.3 Mapping

Mapping is the process of building a coherent model of the environment that clearly and consistently describes the relationships between elements of the environment. The complexity of the model can vary greatly and will be determined by both the working environment and the tasks to be performed. The form of the map can also be influenced by the sensors available: a full three-dimensional map is only possible when ranging (either direct or after processing) is present, whereas an image mosaic-based map (such as that described by Kelly [66]) can be used with little more than a single camera.

Two key uses of the resultant map are to localise the robot and to allow a global path to be planned. Localisation (covered in more detail in Section 2.1.4) typically involves matching a current sensor reading to information in the map or applying estimated motion operations (measured through odometry, for example) to a point in the map representing the robot. A map provides the constraints for many path planning operations (path planning is detailed further in Section 2.1.6). While a straight-line path may be the simplest and most desirable route to take, obstacles and environmental features mean a more complex path is usually required. A useful map will contain information that can be used to find an optimal path (optimal according to some criteria, such as shortest distance or most easily traversed) around intermediate obstacles.

Two basic map creation methodologies have emerged from research: grid-based and topological. Both forms of representation have benefits and pitfalls.

Grid-Based Mapping

Grid-based maps use discretised metric information about the relative physical placement of environmental entities (such as corners and walls indoors or trees and large rocks outside). Early work by Elfes and Matthies [30] and Matthies and Elfes [83] includes the proposal of an occupancy grid that consists of a grid of cells, each of which contains a probability that the region of space at that grid location contains an obstacle. A similar idea is the use of polyhedral geometric representation by Chatila and Laumond [16]. A great benefit of non-uniform cell distribution is that it allows a potentially more efficient representation, as adjacent cells with the same properties can essentially be grouped into a single, appropriately shaped cell. Occupancy grids are a low-level form of representation: there is no inherent link between the probability of occupation of a cell and what exactly is doing the occupying. Typically, anything considered "*non-traversable*" is labelled an obstacle. Extensions can

be made, however, that allow classification or labelling of cells to provide a higher-level of description. Grid-based maps (coupled with appropriate sensors) are well suited to providing a high level of environmental detail, although it is this very fact that means they have the tendency to explode in size in even moderately large-scale environments, resulting in similarly high levels of processing complexity.

Topological Mapping

Topological maps, on the other hand, describe distinct regions or elements as nodes and the relationships between them as edges in a graph representation. For example, Kuipers and Byun [72] presents a topological map that uses "*distinctive places*" and "*local travel paths*" that link nearby places. The qualitative nature of topological maps make them useful for efficiently describing large environments, although there is a central problem associated with robustly and repeatably determining and identifying distinct regions.

When viewed in practical, realisable terms, the distinction between grid-based and topological mapping is often blurry. To be of real use, a topological map will invariably contain geometric information. It could even be argued that the difference is simply one of scale or resolution: a grid-based map could be viewed as a topological one whose nodes are all small-scale, fine-grained features and whose edges are the geometric relationship (perhaps just the distance in a uniform grid or a distance and bearing in a polyhedral configuration).

In more recent work, Thrun [138] proposed an approach that makes use of both paradigms. A grid-based map is partitioned into coherent regions which can then be described by a topological map. The combination of methods aims to harness the advantages of both worlds, resulting in a representation that is accurate, consistent and efficient.

2.1.4 Localisation

Localisation is the process of determining a robot's pose, that is, its position and orientation within its environment. This process is of paramount importance if a robot is to move intelligently through its environment. Clearly, in order to determine how to reach a chosen location, the robot first and foremost needs to know where it is. Localisation methods can be grouped based on whether they use external references (reference-based localisation) or internal measurements (so-called "*dead-reckoning*" methods).

Reference-based Localisation

Reference-based localisation makes use of the known position of external references. These references include active artificial beacons (that emit, for example, light or radio signals), passive artificial beacons (that emit no signal but can be detected by the robot, such as through vision) and natural landmarks (such as trees or hill-tops).

In general, artificial landmarks or beacons can greatly simplify reference-based localisation by providing robust and consistently identifiable markers whose positions can be precisely determined and indeed can be strategically positioned. Clearly, however, the use of artificial references is a limitation: the region being covered must have been visited and prepared first, and the landmarks themselves must be maintained to ensure they are in working order: such preparation and/or maintenance is not always feasible.

Localisation based upon natural landmarks aims to utilise pre-existing environmental features to localise. Methods will require three or more landmarks, as with artificial referencing,

to perform triangulation or trilateration. One of the great difficulties lies in developing methods for robustly and reliably identifying and recognising landmarks, though current work, such as that by Spero [129] has moved towards localisation outdoors using arbitrary environmental features and a multiple-hypothesis data association algorithm.

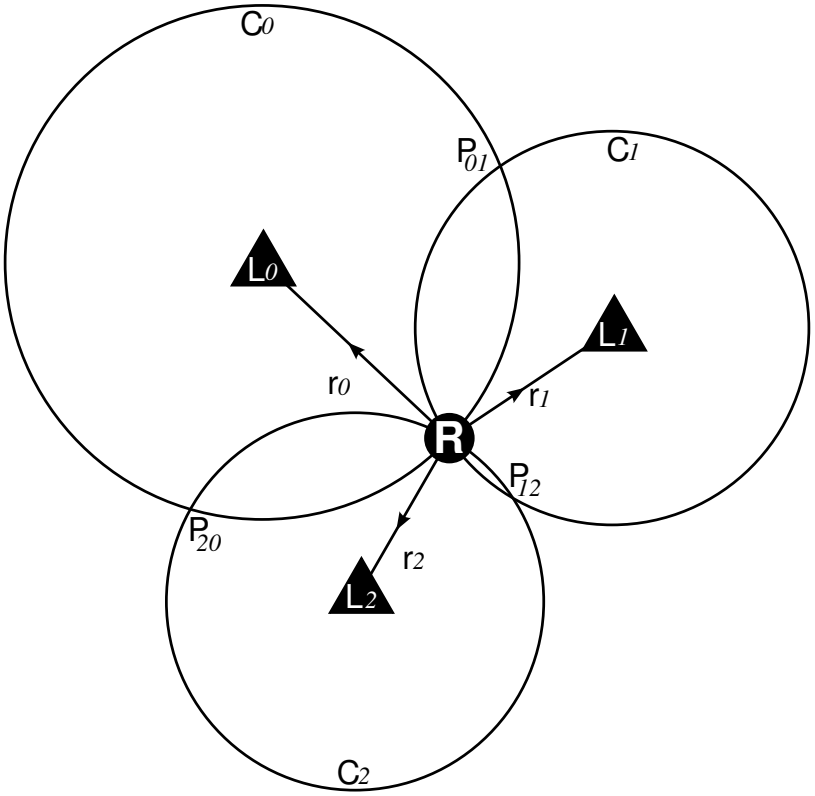


Figure 2.3: Localisation through trilateration

Figure 2.3 gives a graphical overview of the process of two-dimensional trilateration, probably the most common method of terrestrial localisation used when the distances to three landmarks are known. The robot is represented by $\mathbf{R}$ and the three landmarks as $\mathbf{L}_0$, $\mathbf{L}_1$ and $\mathbf{L}_2$ at ranges $\mathbf{r}_0$, $\mathbf{r}_1$ and $\mathbf{r}_2$ respectively. Each landmark and its corresponding range define a circle (for example $\mathbf{C}_0$ corresponding to $\mathbf{L}_0$ and $\mathbf{r}_0$), upon which $\mathbf{R}$ may lie. Knowledge of the range to two landmarks reduces this ambiguity to two points where the two circles intersect. For example, knowing $\mathbf{L}_0/\mathbf{r}_0$ and $\mathbf{L}_1/\mathbf{r}_1$ gives two possible positions where $\mathbf{C}_0$ and $\mathbf{C}_1$ intersect, one at $\mathbf{P}_{01}$ and the other at the correct position $\mathbf{R}$. Knowledge of a third range resolves this further to a single point where all three circles intersect, that is, the correct position $\mathbf{R}$.

One of the most commonly known and widely used reference-based localisation systems is GPS, developed and maintained by the U.S. Department of Defense. An overview of GPS operation is given by The Aerospace Corporation [135] with additional details given by Yinger [158]. Essentially, a group of satellites (active artificial beacons) orbit the Earth and transmit position and timing information to terrestrial receivers. If the signals from multiple independent satellites can be received, the receiver is able to determine its position through trilateration. While GPS has revolutionised the process of global localisation, problems still

exist. A line of sight is required from the receiver to the satellites, meaning GPS can become unusable under or even near heavy foliage cover. Also, while significant improvements have been made to the system (such as the removal of the intentionally degrading Selective Availability [136] and the development of DGPS [150] and WAAS (Wide Area Augmentation System) [35]), position accuracy isn't always good enough for small-scale localisation (such as when position to within, say, tens of centimetres is required). Finally, while a problem for only a subset of robotic research, GPS is very clearly simply inapplicable to off-Earth robotic exploration.

Dead Reckoning

Possibly the most common form of internally-referenced localisation or "*dead-reckoning*" is odometry, typically coupled with the equally popular choice of wheels for locomotion. This usually is performed via wheel encoders that measure either the number of wheel rotations (usually incrementally where, for example, 100 encoder pulses indicates an entire revolution) or for greater precision and control the actual wheel position. Another common internally-referenced localisation method involves the use of gyroscopes and accelerometers to measure inertia and related quantities. Because of the need to integrate measurements to obtain position estimates however, such methods tend to suffer from drift with time and the resultant errors are without bound.

The biggest problem with internally-referenced schemes is the unbounded nature of accumulated errors. These errors can be classed as either systematic or non-systematic. Systematic errors are those associated with the model of the system, for example, a wheel circumference measurement that is imprecise. Errors of these forms are usually more able to be dealt with, for example, through more accurate parameterisation or models involving statistical parameters. Non-systematic errors, on the other hand, are much harder to account for, as they are caused by unpredictable and hence inconsistent variations, such as those that occur through environmental interactions (such as wheel slippage or simple bumps and knocks). A thorough review of work performed in the area of odometry improvement is given below in Section 2.3.1.

2.1.5 Simultaneous Localisation and Mapping

Given the close and often interdependent relationship between the two, it is not surprising that research has been done to find methods of performing mapping and localisation simultaneously. This process, generally referred to as *SLAM* aims to solve the problems associated with trying to localise a robot in an environment when no map, or only a partial map, exists. The SLAM process essentially has a robot building a map while it moves through the environment, all the while using the partial map to localise itself and hence create a consistent, coherent model of its world. The great problem lies with trying to manage three main sources of error: sensor noise (where features are thought to be positioned), navigation error (where the robot is thought to be positioned) and uncertainty in feature correspondence (how currently detected features are matched to those already in a map). Arguably the most commonly used method is that proposed by Leonard and Durrant-Whyte [74] that makes use of an extended Kalman filter to robustly model and account for the three errors. Work has also been done that uses a combination of metric and topological maps [143], particle filters [160, 87] and graph-based maps [38]. In addition, current work by Spero and Jarvis

[128] proposes an alternative method that effectively aims to repeatedly solve the "*kidnapped robot*" problem. (This is the problem of determining a robot's location after an interruption or major discontinuity in the robot's localisation sensing, as would be experienced if a robot were "blindfolded" and taken to arbitrary location some distance from the point of kidnapping.) By doing so, difficulties associated with navigation error are effectively removed from the overall problem.

2.1.6 Path-Planning

Path planning methods aim to determine the route a robot should take to travel from one position to another. Most often there are constraints: travel the shortest distance, use the least energy, or stay out of sight, for example. These constraints must be coupled with environmental elements, such as different ground surfaces, obstacles and undulating terrain, to determine the route to be taken. Additionally, path-planning can occur at different levels of abstraction, from a high-level intelligence system that determines what regions are interesting and worth further investigation and how to get to them, to a low-level reactive system that simply plots paths around immediately detected obstacles. These two extreme examples highlight the differences between *global* and *local* path-planners respectively. Under the guise of Motion Planning, Latombe [73] gives a detailed overview of the many aspects involved in planning the motion of mechanical systems.

Path Planners and Maps

The method used to create and evaluate paths is typically related to the form of map used, reflecting the differing representation of the environment used by each methodology.

Quite possibly the two most common approaches to path planning using grid-based, metric maps are the distance transform and variations [64, 161, 63, 80, 81] and potential fields [51, 1, 96, 42, 107, 108], though other techniques exist such as the application of neural networks [15] and genetic algorithms [57].

The distance transform, as applied to robotic navigation, typically involves a wave-front filling of free space in an occupancy grid with values that correspond to the shortest distance from the goal to the current grid cell. For example, immediate neighbours of the goal will have a value of one, next-nearest neighbours a value of two and so on. Finding a path from any source location (typically the robot's current location) to the specified goal then involves an iterative steepest-descent operation, whereby, starting at the source, a path segment is created from the current grid cell to the neighbour with the lowest value that is less than the current value. In this manner, the path will always move towards the goal, creating an optimal path (in terms of distance), though not necessarily a uniquely optimal path.

Potential field methods create an artificial "*force-field*" throughout the map, typically using two opposing field sources: an attractive source corresponds to the goal and repulsive sources correspond to obstacles. At each point in the map, a resultant force can be calculated from the attraction of the goal and the repulsion of obstacles. A path can then be iteratively created by creating path segments that link the current grid cell with the neighbour in the direction of the resultant force.

Planning with topological maps, on the other hand, typically involves the use of graph theory with Dijkstra's shortest path algorithm (described in many introductory books such as [21]) and varations including those by Saab and VanPutte [121], Fujita et al. [39] being

very common. The A^* search algorithm [52], of which Dijkstra's algorithm is a specific case, is also very popular along with variations such as that given by Trovato and Dorst [147]. These methods use edge properties (such as the corresponding real-world length of the edge and a comparative cost factor used to describe the difficulty of traversing an edge) to find the sequence of nodes and linking edges with the lowest total edge cost from source to destination

Road Following

An important aspect of path-planning in certain environments (importantly in those being investigated) is autonomous road-following. With *a priori* information of the environment, that is, a detailed enough map, and a suitably reliable localisation method, road-following can be considered to be a tightly geometrically constrained general path planning operation. Such detailed information is not always available, meaning alternative methods of following roads are worth investigating. Additionally, even when information is present, the speed of operation of regular and repeated localising operations may not be acceptable, meaning methods that are able to be performed rapidly are able to provide advantages. This leads to the notion of reactive road-following, a local path-planning operation that effectively provides a navigation behaviour. A detailed survey of the current state of road-following research is given below in Section 2.3.3.

2.2 Navigation and the Environment

Robotic operation can be classified based upon the environment in which the robot is to perform. At a high level, an environment can be considered to be either structured or unstructured. Predominantly the focus here is on robots that perform or assist in operations usually done by humans. As a result, the work environment often contains features put in place to assist humans. In differentiating structured from unstructured environments, the defining factor here is primarily the degree to which the environment has been modified to assist operation, specifically navigation and traversal. Accordingly, some arrangement or organised form may exist in what are termed unstructured environments but the degree of structuring is very low when compared to typically structured environments. Details of what constitutes a structured or unstructured environment are given in the following sections, along with brief descriptions of the research performed and the sorts of robots being used in these environments.

2.2.1 Structured Environments

The term *"structure"* relates to the arrangement or organisation of parts within a whole. In the context of an environment and the navigation thereof, we are interested in how the environment has been organised or modified to assist navigation and traversal. Structured environments display a simplicity and consistency that allows them to be easily navigated. Indoors, corridors and walkways are constant widths, walls are straight and generally parallel or perpendicular to one-another, buildings are divided into rectangular regions, doors are evenly spaced. Surfaces have consistent attributes too, whether they be a constant painted colour or a repeated texture or pattern. Outside, things are very similar; roads have constant widths are are predominately straight or smoothly curved, walls and fences are similarly straight or smoothly curved and have constant heights. Surfaces are usually slightly more

irregular than indoors, but we still see consistency in the smoothness of concrete pavements, sealed roads or painted walls, the regular arrangement of bricks or the highly-contrasted and consistent lane markings on roads. A consistent and simple arrangement of the environment means simplifying assumptions and models can be used that make navigation problems easier to solve.

In examining structured environments, it is useful to also make the distinction between robots operating indoors and those operating outside, as the tasks to be performed are often quite different, leading to different approaches in both robots and the algorithms and techniques they use.

Indoor Robotics

Indoor robots work in a variety of environments, including offices, hospitals, homes, laboratories and factories. They include those used for office services like mail delivery and garbage cleanup [113] and automated guided tours [139, 140, 144]. Some of the most recent work centres on the development of humanoid robots (robots that physically resemble people) that will one day perform services relating to household maintenance, healthcare and hospitality. Large companies such as Sony Corporation [126], Honda Motor Co. [55] and Fujitsu [40] have spent a great deal of money developing highly capable humanoid robots that can interact with people in a natural and intuitive manner. A number of commercial products now exist, with perhaps the best known being *Roomba*, the household vacuuming robot from iRobot Corporation [60].

Outdoor Robotics

An area of increasingly active research regarding outdoor robots in structured environments is that concerning intelligent vehicles. This extends from systems that can guide and assist human drivers right through to vehicles that are capable of driving themselves between a starting and target location.

Urban road followers target what is probably first thought of when the word road is mentioned: smooth, paved roads that have clear edges (typically in the form of curbs, gutters or edge markings) and very often have multiple lanes marked with high-contrast painted lines or similar markings. These roads include small single-lane suburban streets through to multi-lane highways and freeways.

Poorly structured rural road following, on the other hand, targets roads that are comparatively degraded. This can mean anything from a lack of clear edges and markings through to gravel or dirt surfaces and boundaries that even a person would have difficulty defining. Roads such as these are often found in less densely populated regions or in areas like national parks, where the environment is intentionally kept as untouched as possible.

Comprehensive surveys of the current state of vision-based intelligent vehicles are given by Bertozzi et al. [4, 5]. In these reviews, the bias towards urban road following research is made clear. Very often such research is motivated by a socially-driven desire for systems that can assist human drivers and increase travel safety and efficiency and in the extreme, systems that allow vehicles to effectively drive themselves. Such a bias has left poorly structured road following a comparatively untouched field of research. A more detailed survey of the current state of road-following research is given below in Section 2.3.3.

2.2.2 Unstructured Environments

As stated earlier, in this context structure is with respect to environmental organisation. Unstructured environments display very little organisation and it logically follows that unstructured environments are almost solely found out-of-doors. An unstructured environment may have become so though degradation, as is the case with collapsed buildings and other such disaster sites. Here, straight and smooth walls have been turned into a mess of rubble and twisted debris. In other situations the surroundings may be natural and essentially untouched, such as deserts and forests. In these places, apart from the odd dirt road, for example, the native flora and fauna have been left to grow and meander and the winds and rain left to shift and shape the terrain, resulting in great irregularity and seeming randomness and a constantly changing landscape. Extreme variants of this are the environments encountered by space-faring robots. The planets and moons of space present many new and unique challenges to robotics and research in general.

Back on Earth, there are two classes of robots operating in unstructured environments that are of particular relevance to the work presented here: Cross-Country Rovers and Urban Search-and-Rescue robots. While the specific workspaces are very different, similarity exists. In both cases, very little is known about the environment beforehand. General descriptions regarding rough, unstable terrain, dark, tight crevasses or wide, open grassy plains are available, but the amount by which the specifics can vary makes the creation of rules and algorithms to navigate very difficult.

Planetary Explorers

Exploratory robots have probably gained most fame through the robots designed to explore other planets, such as the US Sojourner [131] and the Russian Marsokhod [67]. An overview of the history and current state of so-called planetary rovers is presented by Muirhead [89]. Robots such as these are typically designed with manoeuvrability and reliability in mind. Designs often incorporate two or three articulated body segments, each with a pair of independently driven wheels. A configuration such as this is able to traverse rugged terrain and climb over obstacles while maintaining solid contact with the ground and hence maintaining stability. The earliest rover to actually perform off-Earth – on the Moon – was the Soviet Lunokhod [78, 99] which landed in 1970 and was a remote-controlled, wait-and-move type system. Shortly after in 1971 the US landed its Apollo lunar rover on the Moon [155, 14], although this was directly human-controlled and so was not really a robotic system. One of the primary reasons for moving away from teleoperation (i.e. move-and-wait control schemes) towards autonomy came with the development of rovers destined for Mars. With the light round-trip being in the order of 3 seconds for a rover on the Moon, delays between command and actuation were acceptable. On Mars, however, the delays range from 7 to 30 minutes, making wait-and-move unsuitable. The focus of control methodology shifted from full teleoperation to task-based semi-autonomy with humans in more of a supervisory role. Developments were made to create robots capable of autonomously executing behaviour-based tasks that would be chosen and parameterised by human controllers.

Urban Search-and-Rescue

Where exploratory robots are designed to wander and explore largely unknown environments and cross-country rovers are built to cover large distances relatively quickly, search-and-rescue

robots have traditionally they been developed to tackle the problem of exploring a highly confined area, such as that created by a collapsed building. With such tasks in mind, the resulting designs are of little use to other forms of search-and-rescue, especially operations taking place in wide, rugged natural environments. For completeness, however, some of the more prominent search-and-rescue robots are presented here in order to highlight the focus of work so far.

A detailed description of the USAR environment, the applicability of mobile robots to the task and requirements for robot mobility and sensing in USAR conditions is given by Murphy et al. [93]. Two key reasons are given as to why robots should be used in USAR: an USAR site is more often than not a very dangerous place and robots can be made to be expendable, and robots can be made to operate in areas that would otherwise be physically inaccessible to rescuers. Murphy stresses the need for USAR robots to be small and highly agile in order to successfully traverse through the rubble of a collapsed building (a typical USAR scene). The conclusion has been reached that colour video cameras are probably the most useful and versatile form of sensing, in part due to their ability to be used for both autonomous navigation and intuitive teleoperation. Finally, Murphy outlines the value of an adjustable level of autonomy in a robot. For certain situations completely autonomous behaviour is desired whereas in others the robot can be better utilised as a remotely operated tool.

Biological influences on mobile robot design are made clear in Murphy's marsupial and shape-shifting robots [91] and Erkmen's snake robot [31]. Murphy's USAR "Silver Bullet" robot performs the role of marsupial mother to "Bujold", the shape-shifting daughter robot. The marsupial tag comes from the fact that Silver Bullet houses and carries the smaller Bujold, much like a mother kangaroo carries a joey in her pouch. Silver Bullet provides the comparatively delicate Bujold with physical protection, increased processing power and a communication link to remote operators. Bujold is a small tracked robot that can be configured into any of three shapes, each providing a different combination of agility and sensing. One configuration has Bujold laying flat, giving high stability and traction but reduced sensing, as this locates its camera close to the ground. The two other configurations have Bujold sitting upright with the camera facing either forwards or backwards. These provide a better view of the environment at the expense of a significantly smaller footprint and higher centre of mass, reducing stability.

Erkmen's prototype serpentine rescue robot is made up of ten segments with each segment providing one degree of rotational freedom. Movement is provided by a geared servo-motor. By aligning adjacent segments at 90 degrees (rotated along the segment axis), motion in three dimensions is realised. While no mention is made of any ability of the robot to self-reconfigure, there are distinct similarities between this robot and modular, self-reconfigurable robots presented elsewhere (for example in [90] and [157]). As a result, this work is possibly best viewed as an example of the application of an existing design to a particular task. A number of examples and test results of different modes of locomotion are described. The methods include a twisting, side-wise motion, a passive-wheel assisted motion, a rolling-loop motion and a caterpillar-like inching motion.

Cross-Country Rovers

The recent US DARPA Grand Challenges [25], first held in 2004 [23] and then again in 2005 [24], has shown many cross-country roving robots whose primary task is to trek large

distances across the desert autonomously, following a predetermined set of GPS waypoints. Many of these robots took the form of modified commercial vehicles (as opposed to custom-built platforms), essentially cars or trucks heavily modified and equipped with various sensors, actuators and processing ability.

For the 2004 competition, Fulton and Pransky [41] presents a comprehensive overview of the features of various competing robots. Of note is the fact that the top five most successful competitors (ranked by the distance covered by the robot) were all modified commercial vehicles. Of the specified 227 kilometre course through the Mojave Desert, the furthest distance achieved was only 12 kilometres. Additionally, only nine of the twenty-five starters progressed beyond the starting line.

While no entry was deemed to have won the first competition (with the goal being to traverse the 227 kilometre course in under 10 hours), Carnegie Mellon's Red Team Robot Racing [151, 13] entry, "*Sandstorm*", travelled the furthest distance before failure. The team had thirty-five corporate sponsors with published cost estimates of up to US$3.5 million. The robot is a modified Hummer model 998 HMMWV (High Mobility Multi-purpose Wheeled Vehicle), equipped with vision, RADAR, LIDAR and GPS sensors with six-axis inertial sensing and axle encoding being used for position and orientation estimation. A vast amount of geographic and environmental data was collected before the competition, including ultra high resolution/accuracy satellite-acquired elevation images and ground-sourced GPS traces of trails and roads. This data was used in combination with the DARPA-provided GPS competition route to plan a human and computer optimised race route. While the claim is made of "*navigation of unrehearsed terrain*", an extremely large amount of high resolution data was available and used beforehand. The Red Team Racing entry managed to travel 12 kilometres along the course before veering off-course in a hairpin turn and becoming stuck. On attempting to extract itself, the front tyres caught fire and needed to be extinguished by safety personnel. Additionally, the initial collision broke both front axles. It is stated [41] that "*the loss of GPS precision (as vehicle entered steep terrain, blocking part of the GPS constellation) likely contributed to problems in the mountain turns*".

Overall, the success of participants can be described as partial at best. No team managed to complete the course, much less within the allowed ten hours and accordingly, the prize went unclaimed. With the most successful robot travelling barely over five percent of the course length and only slightly more than a third of the starters progressing beyond the starting line, it could be argued that, at least in terms of competition goals, the competition was a failure. However, being the first competition and hence with teams having little if any race experience, unforeseen problems were inevitable.

The second challenge was held in 2005 and took place over a 212 kilometre course, again through the Mojave Desert of the USA. Not surprisingly, given the experience gained through the previous competition, results were much better. Twenty-three vehicles started the race, only one failed to pass the 12 kilometre best result marker from the previous competition, and five vehicles successfully completed the course, four in less than eight hours, comfortably below the ten-hour time limit imposed. Such results are a clear indication that much progress was made by teams between the first and second races.

In a fairly close result, Stanford's "*Stanley*" [130] came first with a time of six hours, fifty-four minutes followed closely by Carnegie Mellon's two entries "*Sandstorm*" (seven hours, five minutes) and "*H1ghlander*" (seven hours, fourteen minutes). "*Stanley*" is based on a Volkswagen Toureg vehicle and contains six Pentium computers, three of which were used simultaneously during the race. Local sensing is performed using five LIDAR scanners, two

radar sensors and a monocular vision system. Vehicle pose estimation is performed using GPS, INS and wheel speed. In addition, among others, a stereo vision system was tested and found to give "*excellent results in the short range, but lagged behind the laser system in accuracy*" [142], suggesting it is a viable alternative to LIDAR for short-range sensing when the superior accuracy of LIDAR is not required. Further details of the Stanford entry are given in [142].

2.3 Navigation Methods and Techniques

In this section we look at some specific navigation methods and techniques that are directly relevant to the work presented in subsequent chapters.

2.3.1 Odometry

This section is further divided into two areas: general attempts to improve odometry measurements, and the application of optical flow to the problem of odometry.

General Odometry Improvements

The problem of overcoming certain forms of odometry error through sensor fusion is investigated by Borenstein and Feng [7]. Their method proposes combining traditional odometry measurement (through wheel encoders) with gyroscope information. The purpose is to detect and correct for impulsive odometry errors (such as those encountered when a platform goes over a sharp bump or knocks into an obstacle) that cannot be accurately modelled through traditional statistical error models. Most of the time their system uses encoders alone for odometry while monitoring gyroscope data. Only when certain events are detected (when gyroscope and encoder readings differ significantly) is the gyroscope data incorporated and used to correct. As a result, it is claimed that the problems of gyroscope drift are essentially avoided and inexpensive gyroscopes, whose drift would otherwise be too large to use continually, are able to be employed. Results presented show that the combined usage of encoders and gyroscope is able to significantly reduce certain forms of non-systematic odometry errors. However, other forms of non-systematic error exist that this method is unable to deal with, with wheel slippage arguably the most important of these for an outdoor robot.

The work presented by Kelly [66], utilises an existing "map" of the environment (an image mosaic) and attempts to estimate a robot's position (that is, perform localisation). It does this by matching a current camera view to an area within the stored mosaic.

Nister et al. [101] presents an alternative system for performing visual odometry. It utilises a feature-tracking front-end (Harris corners are the features detected) that matches point features between frames and creates trajectories by matching features across pairs of frames. These trajectories are then used to estimate the camera pose and hence recover motion estimates. On tests using a stereo vision-equipped platform using DGPS as ground truth, errors of between one and five percent over approximately 200 to 400 metres were reported for translations.

Bunschoten and Kröse [12] describes a method for estimating the translation and rotation of an indoor office robot using an omnidirectional, catadioptric vision sensor. The omnidirectional image is first projected onto a vertically-aligned cylinder giving a panoramic image whose base is parallel with the floor. A Kanade-Lucas-Tomasi feature tracker is used to

track features in this panoramic image across frames and subsequently platform motion is estimated. A method of global trajectory error correction has been suggested, whereby the first and last frame of a trajectory are used to estimate the motion between them and hence correct and close a loop. Clearly this has limited usefulness in a more general setting where points of loop closure are not known and hence knowing which additional frames to compare is difficult if not impossible.

The Application of Optical Flow to Positioning

The application of optical flow techniques to obstacle detection and steering control are described by Lorusso and Micheli [76]. The system uses a forward-facing camera, mounted such that the optical axis is "*nearly*" parallel with the direction of translation. A correlation-based optical flow algorithm was used to calculate the flow and the assumption is made that the ground surface is planar. This assumption is shown to provide good results in terms of angular velocity and "*time to collision*" (related to vehicle velocity) when applied to travel on standard roads. Obstacle detection is achieved by flow-field segmentation based upon either regions of the field with diverging vectors of much higher than average magnitude (for approaching obstacles) or a converging field region (departing obstacles). Results show that the method is able to adequately detect regions containing obstacles that are either approaching or departing. No mention is made, however, of how stationary obstacles are able to be detected, if at all. The assumption of a planar ground surface limits the use of this work to appropriately structured environments.

Lorusso and Micheli [76] and Giachetti et al. [44] describe the use of car-mounted cameras combined with optical flow calculations to measure a vehicle's motion for the purposes of aiding steering control and road navigation. In both cases, the cameras look directly ahead of the vehicle and a complete field of motion vectors is calculated. Lorusso's processing was performed offline on a 60-frame sequence using correlation-based methods. Giachetti describes differential techniques that utilise temporal and spatial derivatives of intensity image sequences, but goes on to state that, because of mechanical noise, a correlation-based method performs better. Using these methods, the optical flow of the image sequences are calculated and used to measure the vehicle's motion and to detect obstacles.

The use of optical flow for estimation of horizontal motion of an underwater robot is investigated by Yu and Caimi [159]. Their method involves a complete optical flow calculation based upon intensity gradients. It is stated that gradient-based methods perform better than full-image correlation-based methods (primarily because of reduced computation) and a limited set of results is provided to support this. It should be noted, however, that while the gradient method is noticeably faster, the difference in accuracy of the two methods is not significant. Results presented show that the notion of recovering motion of a moving platform from image sequences has merit. However, the information garnered is limited; it is stated that "*the developed method assumes that the rotational motion of the subsea vehicle is determined by other means*". No mention is made regarding how a rotation of the platform affects estimations, regardless of whether the rotation can be measured or not.

Nagatani et al. [97] describes the use of a camera looking down at the ground at a 45° angle from an omnidirectional robot, combined with a FUJITSU Tracking Vision image processing board to perform optical flow calculations. Only lateral motion is detected and the results of the optical flow calculations are used to improve the odometry measurements made using wheel-motor encoders. It was stated, however, to be inaccurate compared to traditional

odometry and instead was used in combination with traditional forms of measurement.

What can be taken from this review of the state of the art is that the problem of tracking a robot's motion is still an area worth researching. Problems still exist with traditional methods and while improvements have been made (such as through the fusion of multiple sensors), there is still room for a robust odometry system to be developed that overcomes the problems associated with typical hardware (particularly wheels). The use of vision, and in particular optic flow calculations, has shown to have potential for providing a usable solution if it is appropriately utilised in order to balance precision and operating speed.

2.3.2 Ranging with Passive Stereo Vision

Stereo vision – and related multi-camera range systems such trinocular and multiple-baseline stereo system – has long been a viable alternative to LIDAR and sonar for range measurement, perhaps with the comparatively low equipment costs being a strong motivation. The principles involved have been discussed at length in many textbooks such as Howard and Rogers [56], Xú and Zhang [156], Trucco and Verri [148], Sebe and Lew [125] and the basic theory is well established. In addition, a number of commercial, *black-box* stereo (and trinocular) solutions exist [109, 110] that allow a single unit to be attached to a robot (or any platform) that then provides disparity images directly.

In spite of this, much work has been and continues to be done in developing better algorithms for solving the correspondence-matching problem. An comparison of fast, dense stereo algorithms is given by Sunyoto et al. [134]. A common approach is the application of various forms of dynamic programming [33, 152], though many other forms of improvement have been made including those presented by Hirschmüller [54], Mühlmann et al. [88], Sun [133].

In the context of navigation for robots, stereo vision has been popular for over a decade. Work includes that done by Okutomi and Kanade [102] using multiple stereo pairs with differing baselines, teleoperation using stereo by Jarvis [62], trinocular vision for mapping and path-planning by Murray and Jennings [96], obstacle detection using adaptive colour segmentation and stereo by Batavia and Singh [3], obstacle detection for automotive use by Nakai et al. [98] and wide-baseline stereo in natural environments by Olson and Abi-Rached [103].

2.3.3 Road Following

This section is divided into two areas, the detection and following of structured urban roads and the detection and following of unstructured rural roads. A brief overview of urban road-followers is presented, while more detail is given regarding rural roads and the techniques used to handle the very different conditions.

Structured Urban Roads

Traditional urban road detection and following methods typically make use of features and characteristics of sealed roads that can safely be assumed to exist. Lane markings, distinct and straight edges, locally constant width and planar surfaces are common examples of features exploited.

Common is the use of vision and standard cameras as the primary sensor [71, 77, 162], although variations such as panospheric cameras have also been investigated [20]. Typical

methods involve line fitting, either to detected lane markings [65] or boundaries between the road and surrounds [77]. Such methods, through their relative simplicity have the potential to provide accurate, high-speed road following, but under adverse environmental conditions they can suffer from their reliance on clear and distinguishable features. Alternatively, road texture anisotropy has been examined as a defining road feature [162], where it is assumed that the road surface displays a texture that has a highly anisotropic orientation, compared to surrounding regions that are assumed to appear isotropic. While removing the need for clear markings or boundaries, the underlying assumption is of an artificial road surface (hence the structure in road texture), and so will fail when the road surface becomes degraded and more random in appearance. Once road and non-road regions have been separated, a model is usually used to describe the road in geometric terms. Such models usually make assumptions such as planar road surfaces [162] smoothly curving edges [116, 77] and parallel boundaries [71].

Unstructured Rural Roads

Poorly structured roads are possibly best described as roads and paths that are in various states of degradation. The environments being investigated include dirt roads, mountain passes, winding bicycle paths and degraded rural roads. Sometimes the roads are degraded paved roads, other times they are little more than a path cleared of vegetation. All exhibit the basic and most fundamental structure of all roads, urban or rural: they are relatively clear passageways intended to be used for ground-based travel.

What makes these types of roads different from urban roads is a significant reduction in structure. Markings are typically non-existent, surfaces are often broken, unpaved and irregular and importantly, edges and boundaries are very poorly defined and unclear. The very characteristics that allowed urban roads to be reliably and robustly detected and followed are no longer present and new methods must be applied.

In terms of sensors to be used, cameras and vision remain the most popular choice. In particular, colour images have been commonly used [137, 22, 36, 43] with fair levels of success. Work has been done where colour vision is combined with other sensors, such as LIDAR [114], but even then, the most useful information is gained through vision, with additional sensors providing relatively small improvements, albeit at the cost of increased complexity.

In this different environment, new assumptions must be made. The task of detecting and following a road creates two basic problems: how to extract the road from surrounds and how to model the road to provide useful data. Traditional approaches based on edge-detection make the assumption that firstly, strong, high-contrast edges in the scene correspond to road boundaries or lane markings and secondly, that such boundaries or markings are visibly clear enough to be reliably detected. Under adverse lighting conditions, most notably where strong shadows are present, the most distinct edges are the edges of the shadows themselves, not the road boundaries [137]. The situation is exacerbated when the markings are not present, when road boundaries are poorly defined and vague to begin with or when the road surface is not smooth and is highly irregular, producing many spurious edges. Such conditions cause edge-detection methods to fail.

Instead, the assumption is usually made that the road surface displays some visual characteristic that is significantly different from the surrounding regions, allowing it to be extracted. The most common characteristic is some form of colour representation [137, 22, 36, 43] although the use of texture, either solely [115] or in combination [114] has also been investigated.

Colour-based classifying can be done through off-line training of a supervised learning system [137, 114] or by sampling the actual road to be traversed [36, 43]. While trained systems have the potential to be much simpler (because, ultimately, more is known about the environment beforehand), they suffer from the problem of what happens when the system encounters a situation it wasn't trained for. Either training must be near exhaustive (a prohibitively lengthy if not impossible task) or some fail-safe mechanism must be introduced to deal with such situations. Sampling methods on the other hand, make the assumption that part of the road is visible in a predetermined location (typically an area at the centre bottom of the image). While this allows variations in road appearance to be accommodated, some form of bootstrapping is required (commonly the assumption is made that the robot is initially placed on and in the direction of the road it is to follow) and there is the potential for runaway behaviour if the wrong surface is sampled.

In order to combat illumination problems that arise outdoors, such as shadows and generally inconsistent lighting, various colour models have also been investigated. As well as the standard RGB representation [137, 22], variations on HSI have been used. A common approach [36, 43] is to use a reduced H/I representation, where hue is used unless either saturation or intensity is low, in which case intensity is used instead. This is intended to overcome purported instabilities that occur with hue at low saturation or low intensity. There are potential problems with this approach, however, as shadows have low intensity and will be represented by intensity, not hue like surrounding road regions. As a result, the potential benefits of using a colour model that can filter out illumination variations is at least partially lost. Dominant texture orientation has been used [115] in extreme cases where there is almost no discernible difference between the intended road and surrounding areas, such as on dirt roads running through a desert. Here, bands caused by previous vehicle traversal leave a texture with a dominant orientation that can be used to estimate the road's vanishing point.

Processing based on the above assumptions usually produces a modified image that contains regions that are labelled as road or non-road, either distinctly or probabilistically. From here, a constrained road model is typically applied that firstly refines the definition of what constitutes the road and secondly can be used to provide useful information, such as approximated road boundaries, steering commands or road vanishing points. Here, assumptions about road geometry are made. Common assumptions include constant road width [137, 36], straight road edges [137, 22, 43], and near parallel boundaries [36]. Models can be simple like a triangular road model that assumes straight road edges and a fixed width [137]. More restrictive models include an image-based road model consisting of predetermined road, shoulder and non-road regions [36]

This review shows that the majority of work on following roads has focused on structured urban roads that exhibit very different characteristics to poorly structured rural roads. The techniques that have proven to be valuable with structured roads (most notably edge detection and line detection) are simply not reliable when tackling rural roads. The most useful methods regarding rural roads involve the use of colour vision and some form of region classification, although a robust and adaptable classifier is still far from having been perfected. Additionally, the reliance on relatively rigid models of the road, and the underlying assumptions about the road's structure, mean there is room for an improved solution if such assumptions and hence limitations can be effectively relaxed.

2.4 Revisited: The Problem of Visual Navigation on Rough Tracks in Bush Environments

There are two key problems associated with navigation in bush environments that are being tackled here: relative localisation for motion control and local path-planning. In addition, as it is an important part of a complete navigation system, the problem of global path-planning is also being investigated.

Relative Localisation: The environments being investigated are predominantly natural. A consequence of this is that the ground surfaces encountered are typically irregular and more importantly slippery. The level of traction provided to a mobile robot can vary considerably, even when the platform is moving over areas intended for traversal – that is, along tracks – leading to wheel slippage and hence discrepancies between perceived motion (when traditional wheel encoders are used) and actual motion. While solutions exist, in the form of separate driven and encoder wheels, that mean driven wheel slippage does not produce errors, such solutions are typically both bulky and mechanically complex, especially if they are to work in all directions of translation. The proposed solution uses only passive vision, with hardware requirements of no more than a camera or two. The complexity of a mechanical solution and the errors associated with such complexity can then be avoided and a more robust solution obtained.

Local Path Planning: The immediate problem being addressed is that of determining "*amenable terrain*". Many parts of the terrain may be traversable in the physical sense – a grassy field is potentially traversable for example – but there are areas, within the target environment, over which we would prefer to travel: dirt roads and tracks. We propose here to determine what is "*preferably traversable*" in terms of two feature-spaces. The first is the space of obstacles. Obstacles serve to physically restrict the choice of robot path by presenting areas – that perhaps but not necessarily belong to distinct objects – that would be potentially unsafe to travel into or over. The second space is that of the road surface. This represents the areas that we believe are both safest and most beneficial to travel over. Obstacle-space can be thought of as a negatively influential field while road-space is positively influential: we wish to stay away from obstacles but on the road. Initially these two spaces will be sensed and processed independently resulting in probability or likelihood maps indicating the confidence that an area can be safely traversed. Combining these maps we can then arrive at a more complete description of the environment, from which a local path can be planned that serves to take us along the road while avoiding obstacles.

Global Path Planning: The final element gives a high-level context to the problem. Bush environments have very often already been mapped by various local authorities and surveying companies, and it makes great sense to make use of such knowledge where possible. Accordingly, a global path-planner is proposed that can use such maps to create hybrid maps of the environment suitable for planning robot paths. The proposed maps use metric locations, typically corresponding to intersections or places of interest, and topological edges, corresponding to segments of road that connect locations. Using metric locations allows map nodes to be described unambiguously *a priori*, given that most if not all maps have grid coordinates that can be converted to a consistent frame of reference (such as latitude and longitude). Addi-

tionally, the use of metric locations provides a means for dynamically producing cross-country paths when required without further knowledge of the terrain.

The use of passive vision: In addition to the benefits of using vision for sensing outlined in Section 1.4.2, passive vision was chosen for sensing in order to determine whether passive vision alone could be used to achieve the task. Throughout history, many engineering problems have been solved by simply using a large-enough set of resources. The wide range of sensors used on most if not all DARPA Grand Challenge entries (most had at least multiple vision sensors and LIDARs, many had RADAR systems in addition) is one such example of this approach to problem-solving. Here we instead choose to use what is believed to be a minimal but sufficient sensor set. Perhaps the biggest potential problem with vision is the processing required to turn a raw image into usable information. If it can be shown, however, that vision coupled with the processing hardware available today can provide an effective solution, then clearly a future implementation of that solution, using what will undoubtedly be superior computing power, will result in superior performance.

Chapter 3

Relative Localisation with Visual Odometry

> “ *We're not lost. We're locationally challenged.* ”
>
> **John M. Ford, 1957–2006**

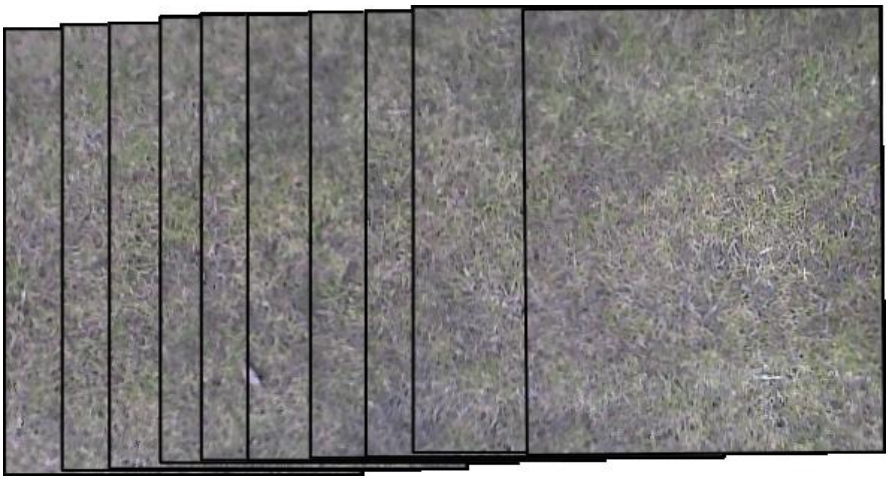


Figure 3.1: A sequence of images, taken from a moving robot and stitched together

In order to navigate with purpose, a mobile robot needs the ability to localise, that is, the ability to determine its position and orientation (together often referred to as a pose) within its working environment. This process typically equates to either directly determining the robot's pose (for example, through the use of GPS) or measuring the robot's motion (that is, a change in pose) and integrating this to determine each new pose. In the case of a teleoperated robot, a human operator will invariably observe the robot as it moves through its environment and respond accordingly, providing the feedback necessary for controlled motion. An autonomous or semi-autonomous robot, however, must be able to sense and measure its own movement.

3.1 Introduction

Localisation is the process of determining a robot's pose within some frame of reference, usually one consistent with the specification of the robot's workspace. A number of methods and mechanisms for achieving this were discussed in Section 2.1.4. The term "*relative localisation*" as used here refers to the process of finding the robot's pose with respect to a previous pose. Alternatively, it can be viewed as the process of estimating the robot's motion by measuring a change in pose. The process can be divided based upon whether the measurements used to localise observe external features (leading to reference-based relative localisation) or internal features (leading to what is commonly termed "*dead-reckoning*").

Dead-reckoning methods are highly useful because they are typically high speed and independent of recognisable external features. Arguably, the most common form (in no small part because of the popularity and relative maturity of wheeled platforms) is wheel-encoder-based odometry. The principle is exceptionally simple: count the number of wheel rotations (or parts thereof) and multiply by the wheel circumference, resulting in a measure of how far the wheel (and hence robot) has travelled. Multiple encoders (commonly one on each of two differentially driven wheels) can be used to measure the platform's rotation and distance and hence track a robot's complete motion. There is one major issue, however, with the use of dead-reckoning: in measuring only internal properties, we are making the implicit assumption that there is a definite and predictable relationship between the measured internal properties and motion through the environment. Much of the time this assumption holds true: the robot turns a wheel and it moves. Sometimes, however, this fails, whether because the robot turned a wheel but didn't move (because the wheel slipped) or because the robot didn't turn a wheel but moved anyway (perhaps the robot skidded on a patch of wet floor). Further sources of error are passive in nature (that is, they are not the direct result of a difference between requested and actual action). Errors arise from poor calibration (such as an incorrect wheel circumference being used), wheel distortion (whether through wear or deformation under load) and contact variations during turning. Often (and preferably when possible), the wheels used for odometry are separate from the wheels being driven. This can reduce or remove problems associated with drive (that is, most slip and skid errors), but cannot counter passive errors. Whatever the cause, such failures introduce unpredictable errors that accumulate without bound and limit the usefulness of dead-reckoning for localisation.

Reference-based methods perform localisation with respect to external, environmental features that are usually implicitly assumed to be stationary. In the case of relative localisation, the absolute locations of these features need not be known. It is the location of features *relative to* the robot that is used. These methods are highly useful because they are resilient to errors in action. That is, if a robot is given a command to move and it does not move, because external environmental features are the reference points, a correct measurement of no motion is made. A problem with many forms of reference-based localisation is the reliance on external feature extraction and analysis. Because these reference-based methods rely upon measuring the position of features external to the robot, the problem often becomes one of being able to reliably and repeatedly find and identify suitable features. In some environments this is simplified through the use of artificial landmarks (see Section 2.1.2) that have been placed specifically for the purpose of navigation. This is not always possible, however, especially in large-scale outdoor environments that cannot effectively be regularly maintained. More generally, often-complex preprocessing must be performed to extract and match usable natural features from raw sensor data.

Somewhere between these two forms of localisation are systems that uses RADAR-based ground-speed sensors. With the sensors angled down towards the ground, these systems use Doppler-shift principles to measure the speed of the ground, relative to the sensor. Work using these systems includes the use of two sensors to measure slippage [149] by measuring ground speed with one sensor and wheel speed with a second sensor, and the use of dual sensors, operating primary as a differential pair, in combination with DGPS to improve heading estimation [117]. The sensors themselves, such as those available from DICKEY-John Corporation [27] are commonly used to measure the true speed of agricultural vehicles that often work on soft, slippery ground. A single sensor is only able to give an estimate of speed – not velocity – and so multiple senors must be used to produce a complete (two-axis translation and rotation) estimate of motion, adding to cost, complexity and calibration requirements.

The visual odometry system described in this chapter aims to provide a robust and reliable relative localisation mechanism that is externally referenced but can use raw sensor data, eliminating the need for costly feature extraction. The basic setup involves the use of a camera, rigidly mounted to the robot, that observes the ground. When the robot moves (regardless of the cause of motion), the view of the ground will also move, and that motion can be detected and measured, giving an estimate of the robot's movement. Figure 3.1 shows a sequence of images captured while the robot was moving. Overlap between sequential images means the motion between images can be recovered. In this figure, instead of simply recovering motion, the images have been stitched together, showing the path taken.

3.2 Outdoor Environments: A Curse and a Blessing

Outdoor environments are inherently less predictable than indoor environments. It can be much more difficult – and sometimes impossible – to regulate many aspects that we would like to. Natural environments are often at the most difficult end of the scale to control. As a simple example, wind causes trees to move, which in turn cause their shadows to move, which in turn means the patch of ground upon which their shadows fall changes in appearance repeatedly in a matter of seconds. Ground surfaces display similar unpredictability because they are typically formed from irregular and multiple sources of material. Even regions such as gravel roads, passageways specifically designed and prepared for predictable use, exhibit irregularity, evidenced by the need for those driving along them to be far more careful than would be necessary on controlled and maintained sealed roads.

One of the big problems faced by a mobile robot outdoors in essentially natural terrain is that of traction, or, more specifically, lack or uncertainty thereof. It is very easy, whether the ground is wet mud, dry sand, grass or gravel, for a robot (or indeed, any other vehicle) to attempt a movement and end up going no-where or where not intended. This is a problem for all forms of locomotion: hiking boots and off-road tyres have the same chunky tread patterns because they both seek greater traction on rough and slippery surfaces. The problem of poor traction applies to the reverse situation: on a slope, for instance, a robot that thinks it is stationary (because it has halted its motors), could in fact be sliding.

This irregularity can, however, be seen as both a curse and a blessing. It creates an unpredictability that can make control very difficult, but it also provides a uniqueness that can be used to help solve this problem. The irregularity can show itself in a visual form through the great variety of textures and almost-patterns (almost, because things don't quite seem to repeat themselves exactly) that are seen all through the environment. Of particular

interest is the high degree of non-repeating texture on most if not all outdoor ground surfaces in natural environments. This richness opens up the possibility of using computer vision to firstly uniquely characterise parts of the immediate environment (that is, patches on the ground near the robot) and then find these same patches in subsequent images, thereby providing the information necessary to estimate the robot's motion.

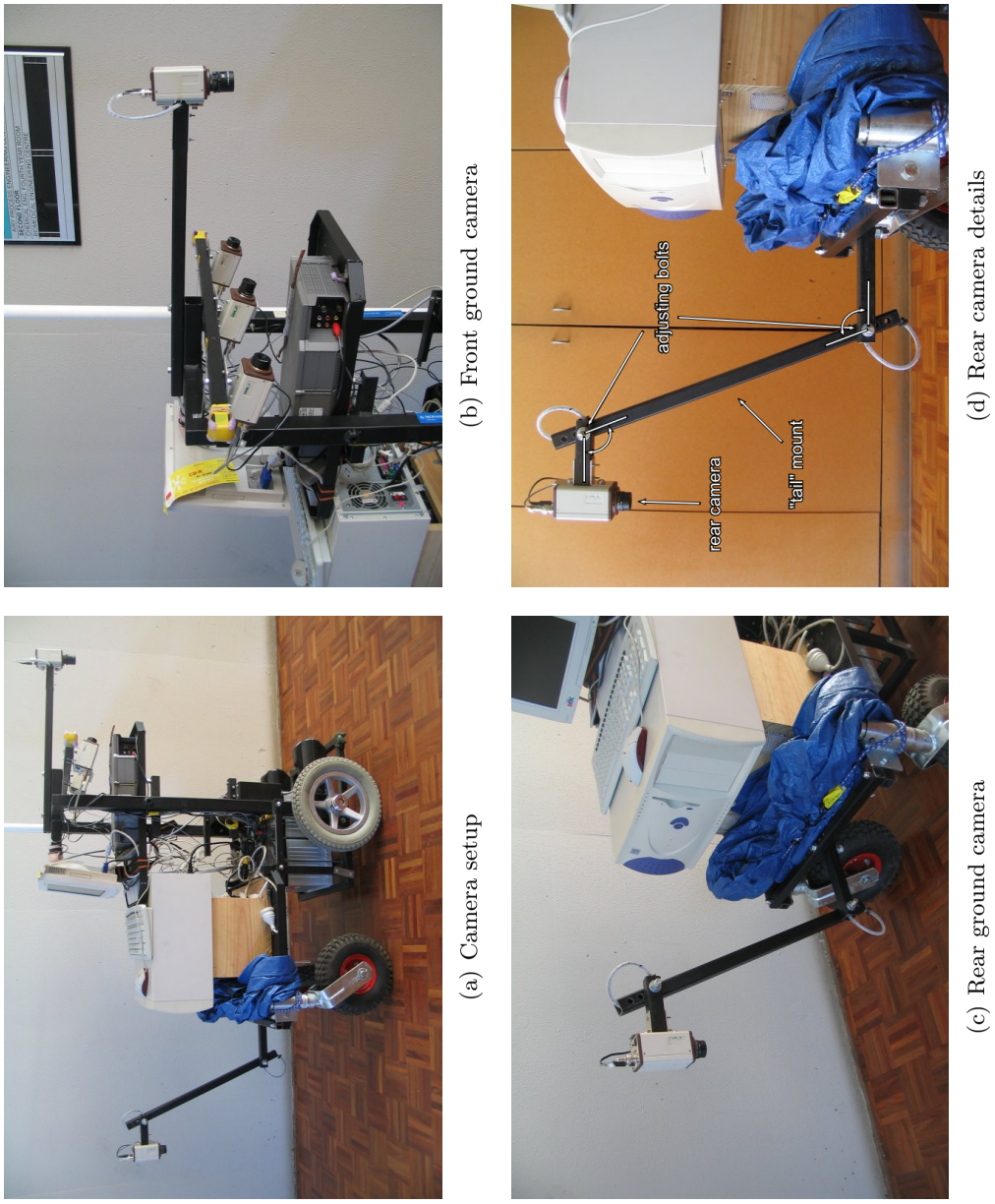
3.3 Hardware Setup

The basic hardware requirement for a visual odometry system is simply a single camera, rigidly mounted to the robot and oriented to face the ground. In the setup used, a camera is unable to be mounted near enough the robot's centre of rotation (a point between the two differentially driven wheels) while maintaining a clear view of the ground. As such, when the robot is rotating, the motion recovered from a single camera is essentially a tangential translation. For improved performance with regards to rotation estimation, multiple cameras can and should be used. By using two cameras, mounted at opposite ends of the robot (one on either side of the likely centre of rotation), rotations can be estimated with much greater reliability. While the cameras could be mounted in a variety of ways (such as at the end of beams directly above the driven wheels' axles), it was decided to mount them in a front-rear configuration in order to keep the robot relatively narrow (and hence able to be manoeuvred through narrower gaps). Two cameras provide an improvement because, for example, under clockwise rotation (viewed from above), a camera at the front will perceive motion to the right while a camera at the rear will perceive motion to the left. It is this differential motion that then allows rotations to be estimated effectively.

In using multiple cameras, we effectively treat the images acquired by each camera as being isolated regions of a larger image. Calibration gives the position of each camera relative to a common, robo-centric frame of reference so that the coordinates of a point in either image can be transformed into a single, common frame of reference.

The two cameras used are standard $\frac{1}{3}-$inch CCD video cameras with a stated sensor resolution of 537×597 pixels, producing an analog PAL signal. Each is housed in a rugged security-style metal case and uses CS-mount lenses. For the front camera, the lens used is of 6.0 mm focal length, giving an effective field-of-view of approximately 44 degrees horizontally and 33 degrees vertically. For the rear camera, the lens used is of 4.0 mm focal length, giving an effective field-of-view of approximately 62 degrees horizontally and 47 degrees vertically. Different focal lengths are used to partially counter the difference in views caused by the cameras being mounted at different heights. Calibration and conversion of image locations to world locations, however, mean the use of different focal lengths is in no way detrimental. Each camera is coupled to standard PC frame-grabber, specifically a FlyVIDEO 3000 PCI capture card, which uses the Philips SAA7134 chipset. Images of 320×240 pixels are used.

The front camera is mounted at the end of a beam on top of the robot, as shown in Figure 3.2(b). It is positioned so that none of the robot is visible in images. The rear camera is mounted on a configurable but rigid "tail", as shown in Figure 3.2(c) and Figure 3.2(d). In addition to the rotational degrees of freedom afforded by the adjusting bolts (as shown in the figure), the tail can slide further out to position the camera further away from the platform if necessary. Once positioned and oriented as desired, the mount is made rigid simply by tightening all bolts.



(a) Camera setup (b) Front ground camera

(c) Rear ground camera (d) Rear camera details

Figure 3.2: Visual odometry camera mounts

3.4 The Geometry of Visual Odometry

3.4.1 Assumptions

Visual odometry makes a small number of underlying assumptions about the geometry of the world and robot. These assumptions and their consequences are described as follows:

Firstly, it is assumed that the visible ground is relatively flat, specifically that the range of scene depth (the difference between the closest visible point and the furthest visible point) is small compared to the average scene depth. This means that there is zero or negligible parallax error and, accordingly, any small patch of the ground seen in an image appears essentially the same (except for movement in the image frame) before and after motion and hence can be tracked. This process is, of course, limited to tracking patches based on image fragments which are visible in both image frames.

Secondly, it is assumed that the robot's motion is restricted to translations along the ground plane and rotations about an axis perpendicular to the ground plane. This is a simplifying assumption that reduces the motion to three degrees of freedom and is applicable to most terrestrial robots (exceptions being some legged robots that experience significant vertical motion and snake-like robots that have many more degrees of freedom).

Finally, it is assumed that the ground is locally planar, such that the ground seen in images lies on the same plane as the robot itself. To recover robot motion, points in the image are projected onto the ground in a world reference frame. To simplify the process and remove the need for dynamic depth information (such as would be obtained with a stereo system, for example), the system is calibrated with the robot sitting on flat ground and all points are assumed to lie on this plane.

3.4.2 Describing Motion

Consider a point on the ground (defined as the $z = 0$ plane), expressed in normalised homogeneous coordinates as $\boldsymbol{p} = [x, y, z, w]^T$. Normalisation sets $w = 1$ while the ground condition sets $z = 0$. Under rigid motion, the point's movement from $\boldsymbol{p}$ to $\boldsymbol{p'}$ can be described as a rotation R followed by a translation T. Assuming that the rotation is by an angle θ about the z-axis (in a right-handed manner such that, when looking along the positive z-axis towards the origin, a rotation by a positive angle is counter-clockwise) and the translation by dx in the x-direction and dy in the y-direction, the motion can be described in matrix notation as

$$\boldsymbol{p'} = \mathsf{TR}\boldsymbol{p} \tag{3.1}$$

where

$$\mathsf{T} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } \mathsf{R} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

More compactly, we can express this as

$$\boldsymbol{p'} = \Phi\boldsymbol{p} \tag{3.2}$$

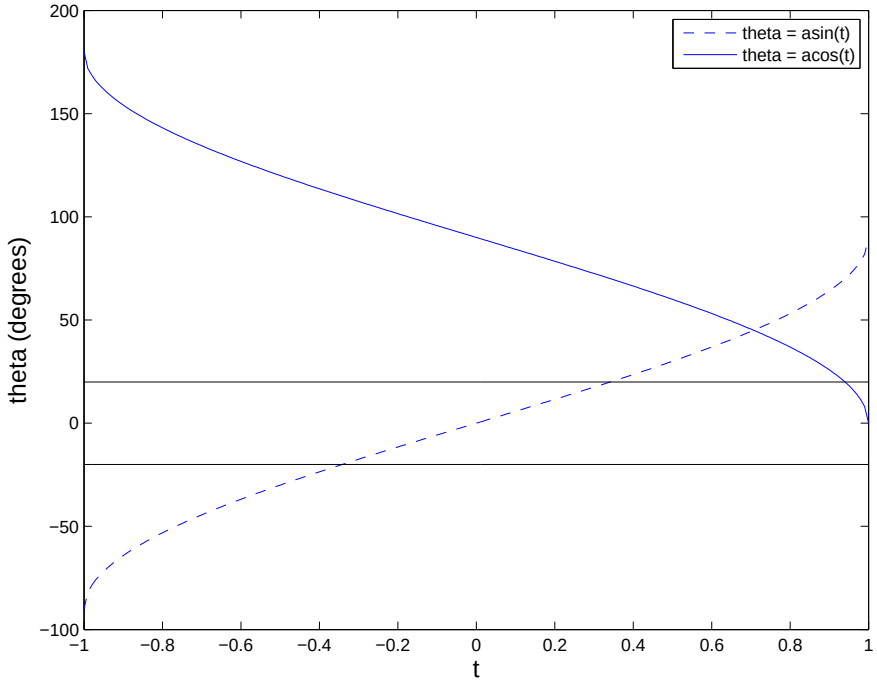


Figure 3.3: Effects of errors using inverse trigonometry

with

$$\Phi = \mathsf{TR} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & dx \\ \sin\theta & \cos\theta & 0 & dy \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

Note that, while there are only three actual parameters of motion (θ, dx and dy), we have in our linear equations four unknowns: $\sin\theta$, $\cos\theta$, dx and dy. Ideally, we would recover the same θ from both $\sin\theta$ and $\cos\theta$, but errors in measurement of $\boldsymbol{p}$ and $\boldsymbol{p'}$ mean they will most likely differ. For small angles, θ as calculated from $\cos\theta$ is far more sensitive to errors in $\cos\theta$ than θ as calculated from $\sin\theta$ is to errors in $\sin\theta$. This is shown graphically in Figure 3.3. For angles between -20 and $+20$ degrees (between the solid horizontal lines), variations in t produce much larger variations in θ when calculated from $\mathrm{Cos}^{-1}t$ compared to $\mathrm{Sin}^{-1}t$. Therefore, we choose to ignore $\cos\theta$ from final calculations and instead determine θ from $\sin\theta$ alone.

To solve for these four unknowns, we need four equations, with each $\boldsymbol{p}, \boldsymbol{p'}$ pair giving two equations. The transform of two such points, $\boldsymbol{p_1} = [x_1, y_1, 0, 1]^T$ and $\boldsymbol{p_2} = [x_2, y_2, 0, 1]^T$ gives the set of four equations

$$\begin{array}{rcccl} x'_1 & = & \cos\theta \cdot x_1 - \sin\theta \cdot y_1 + dx & = & x_1 \cdot \cos\theta - y_1 \cdot \sin\theta + dx + 0 \\ y'_1 & = & \sin\theta \cdot x_1 + \cos\theta \cdot y_1 + dy & = & y_1 \cdot \cos\theta + x_1 \cdot \sin\theta + 0 + dy \\ x'_2 & = & \cos\theta \cdot x_2 - \sin\theta \cdot y_2 + dx & = & x_2 \cdot \cos\theta - y_2 \cdot \sin\theta + dx + 0 \\ y'_2 & = & \sin\theta \cdot x_2 + \cos\theta \cdot y_2 + dy & = & y_2 \cdot \cos\theta + x_2 \cdot \sin\theta + 0 + dy \end{array}$$

which can be expressed as

$$\boldsymbol{q'} = \mathsf{Q}\boldsymbol{\phi} \tag{3.4}$$

where

$$\boldsymbol{q'} = [x'_1, y'_1, x'_2, y'_2]^T \tag{3.5}$$

is a vector containing the combined relevant $\boldsymbol{p'_1}$ and $\boldsymbol{p'_2}$ coordinates,

$$\mathsf{Q} = \begin{bmatrix} x_1 & -y_1 & 1 & 0 \\ y_1 & x_1 & 0 & 1 \\ x_2 & -y_2 & 1 & 0 \\ y_2 & x_2 & 0 & 1 \end{bmatrix} \tag{3.6}$$

is the matrix containing the relevant coordinates of $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$, and

$$\boldsymbol{\phi} = [\cos\theta, \sin\theta, dx, dy]^T \tag{3.7}$$

is a vector of the unknown transform parameters.

If $\boldsymbol{p'}$ and $\boldsymbol{p}$ (and hence $\boldsymbol{q'}$ and Q) are known, 3.4 can be solved for $\boldsymbol{\phi}$ and the rotation angle and translation motion vector can be found.

3.5 Motion Estimation from Images

Having described how the displacement of two points can be used to recover a rotation and translation, we now look at how the points themselves are defined and how their displacement can be found.

3.5.1 Description of Features

Outdoors, especially in what are essentially natural environments (except for tracks), there exists great irregularity. While this can sometimes be a burden, it can also sometimes be a great help. In using an external reference to perform relative localisation, we require features that are unique enough to be distinguished from other potential features in the vicinity. Unlike in man-made environments, the ground in bush environments, especially on a small scale, displays a high level of visual irregularity which means points on it can be characterised and matched efficiently and reliably. Specifically, we can simply use image patches as features: the irregularity in the appearance of the ground means that there is a high probability that we will be able to find these same patches uniquely in a subsequent image.

A feature is then described by a patch of an image, which we refer to as a template τ, defined by a centre point $[r_\tau, c_\tau]^T$, (row and column numbers) a width of $2C_\tau + 1$ image columns and height $2R_\tau + 1$ image rows.

In the mathematics presented above, we require two points to resolve motion into a rotation and translation (see Equation 3.4). For a more robust estimation, we can use a greater number of points. Given the lack of prior knowledge and the somewhat arbitrary nature of ground images (other than knowing we are viewing the ground, we do not know in advance what the ground will look like or where we should look for features), we choose to simply use a uniform grid to define the locations of our features. We will then look for these features in the next sequential image.

3.5.2 Matching Patches

With a feature being defined simply as an image patch (template), we now focus our attention on the problem of locating (by matching) a given patch in an image, that is, the template matching problem.

To find the new location of a given template, we define a search window ω, defined by its centre $[r_\omega, c_\omega]^T$ (typically equal to $[r_\tau, c_\tau]^T$), width $2C_\omega + 1$ and height $2R_\omega + 1$. We then perform a similarity operation, finding the similarity of the template with an equivalently sized patch for each possible displacement within the search window. The displacement giving the best similarity measure is deemed to be the new location of the template.

Specifically, for a generalised similarity function $A \circledast B$ between a template τ in image A and an equally-sized patch in image B centred at $[r', c']^T$, correlation-based similarity is defined as

$$S(\tau, r', c') = \sum_{r=-R_\tau}^{R_\tau} \sum_{c=-C_\tau}^{C_\tau} A[r_\tau + r, c_\tau + c] \circledast B[r' + r, c' + c] \tag{3.8}$$

which can also be area-normalised as

$$S_\eta(\tau, r', c') = \frac{S(\tau, r', c')}{\eta} \tag{3.9}$$

with normalising factor

$$\eta = \sum_{r=-R_\tau}^{R_\tau} \sum_{c=-C_\tau}^{C_\tau} |A[r_\tau + r, c_\tau + c]| \times \sum_{r=-R_\tau}^{R_\tau} \sum_{c=-C_\tau}^{C_\tau} |B[r' + r, c' + c]| \tag{3.10}$$

The template-matching problem consists of finding the $[r', c']^T$ that either minimises or maximises (depending on the specific function $A \circledast B$) the function S, subject to the window constraints:

$$r' \in \{r_\omega - R_\omega, \cdots, r_\omega + R_\omega\}$$
$$c' \in \{c_\omega - C_\omega, \cdots, c_\omega + C_\omega\}$$

Measures of Similarity

There are a number of possible choices for $A \circledast B$, with perhaps the three most common being the absolute difference

$$A \circledast B = \Big|A - B\Big| \tag{3.11}$$

to be minimised, the squared difference

$$A \circledast B = \Big(A - B\Big)^2 \tag{3.12}$$

to be minimised, and the product

$$A \circledast B = A \times B \tag{3.13}$$

to be maximised.

In addition, mean-normalised versions can be used that substitute $\widehat{A}$ for A and $\widehat{B}$ for B

where

$$\widehat{A} = A - \bar{A} \tag{3.14}$$
$$\bar{A} = \frac{1}{RC} \sum_{r=0}^{R-1} \sum_{c=0}^{C-1} A[r, c]$$

and similarly

$$\widehat{B} = B - \bar{B} \tag{3.15}$$
$$\bar{B} = \frac{1}{RC} \sum_{r=0}^{R-1} \sum_{c=0}^{C-1} B[r, c]$$

with R and C being the common dimensions of the patch in A and B.

The main reason for using mean-normalised similarity measures is to compensate for variations between A and B that would result from lighting changes (typically due to variations in viewpoint) or differences in camera gains (if A and B were taken from difference sources). For example, with A and B being identical save for a change in the average intensity, the unnormalised difference measures will produce an overall difference that is the product of the patch area and the difference in average intensities. In the context of visual odometry, both images come from the same device and will be separated by a small time difference. Therefore, it is highly unlikely that intensity differences will be present that would erroneously bias results. As such, mean-normalised measures of similarity needn't be considered further, leading to a computational saving.

In comparing the measures in Equations 3.11, 3.12 and 3.13, we are looking for a function that both performs well and is computationally efficient (that is, is fast). When normalised (using Equations 3.9, 3.14 and 3.15), cross-correlation (Eq^n 3.13) performs very well under changes in intensity and contrast. With negligible intensity variation across images assumed, however, the benefits of normalised cross-correlation are lost. Under these assumptions, 3.11 and 3.12 perform well and require no costly normalisation. As such, we can remove normalised cross-correlation from further consideration.

Additionally, because 3.11 and 3.12 are measures that need to be minimised, they present a great opportunity for optimisation. As stated, the best match is the one that produces the greatest similarity, in this case the smallest difference. Clearly, for a given A/B pair, for it to be the best it must produce a smaller difference than all previous A/B pairings. We can use this to speed up calculations by using an exit or jump-out condition whereby we stop the summation if it exceeds the current lowest difference, as, in either 3.11 or 3.12, the sum can only get larger if we continue (as we are summing values that are zero or positive). We therefore avoid excess calculations when we know we cannot possibly obtain a better match. With 3.13 needing to be *maximised*, optimisations of this form are not possible.

In comparing the remaining measures 3.11 and 3.12 we note that the absolute-difference and squared-difference measures perform very similarly and as a result, absolute-difference is chosen for it's slightly greater computational efficiency (an absolute operation compared with a multiplication).

The algorithm used for matching N_F features from image I_{i-1} in image I_i is shown in Algorithm 3.1.

Algorithm 3.1: Visual Odometry feature matching algorithm

Procedure: P' = `MatchFeatures(`P`,` I_i`,` I_{i-1}`)`

Input: current image I_i and previous image I_{i-1}
Input: set of N feature points $P = \{\boldsymbol{p_0}, \ldots, \boldsymbol{p_{N-1}}\}$ from image I_{i-1}
Output: set of N matched feature points $P' = \{\boldsymbol{p'_0}, \ldots, \boldsymbol{p'_{N-1}}\}$ from image I_i

Data: ROWS, COLS ▷ image dimensions
Data: R_τ, C_τ ▷ template dimensions
Data: R_ω, C_ω ▷ window dimensions
Data: d, d_{best} ▷ difference, best difference

1: $P' \leftarrow \{\emptyset\}$ ▷ initialise matched feature set
2: **for** $\boldsymbol{p} \in P$ **do** ▷ process every feature in the set
3: $(r_\tau, c_\tau) \leftarrow \boldsymbol{p}$ ▷ get the next feature location
4: $d_{best} \leftarrow \infty$ ▷ initialise the best difference for this feature
5: **for** $r' \in \{r_\tau - R_\omega, \ldots, r_\tau + R_\omega\}$ **do** ▷ for every row in the window
6: **for** $c' \in \{c_\tau - C_\omega, \ldots, c_\tau + C_\omega\}$ **do** ▷ and for every column
▷ incrementally calculate the sum of differences
7: $d \leftarrow 0$
8: **for** $r \in \{-R_\tau, \ldots, +R_\tau\}$ **do**
9: **for** $c \in \{-C_\tau, \ldots, +C_\tau\}$ **do**
10: $d \xleftarrow{+} \left| I_{i-1}[r_\tau + r, c_\tau + c] - I_i[r' + r, c' + c] \right|$
11: **if** $d > d_{best}$ **then** ▷ if difference is worse than the best
12: **goto** *19* ▷ stop calculating
13: **end**
14: **end**
15: **end**
16: **if** $d < d_{best}$ **then** ▷ if difference is better than the best
17: $d_{best} \leftarrow d$ ▷ store the new best match
18: $\boldsymbol{p'} \leftarrow (r', c')$
19: **end**
20: **end**
21: **end**
22: $P' \xleftarrow{\cup} \boldsymbol{p'}$ ▷ add the best match to the set of matched features
23: **end**

▷ P' now contains the best feature matches

3.6 Practical Considerations

In the following sections we discuss the more important practical issues involved with the implementation and use of a visual odometry system. First we look at the use of multiple features to provide a better solution and the need to remove outliers. We then discuss the system parameters and how they effect the performance of the system, including how they can be adjusted to suit a specific problem. Finally, we look at how the hardware choice and setup can affect the results and how this can be used to provide a better problem-specific solution.

3.6.1 The More the Merrier

From the mathematics presented above we can see that the previous and current locations of two features are required to resolve motion into a rotation and a translation. In order to minimise errors, however, more features can and should be used. In the general case using n features, Equation 3.4 can be modified to

$$\boldsymbol{q'} = \mathsf{Q}\boldsymbol{\phi} \tag{3.16}$$

with

$$\boldsymbol{q'} = [x'_1, y'_1, \ldots, x'_n, y'_n]^T$$

$$\mathsf{Q} = \begin{bmatrix} x_1 & -y_1 & 1 & 0 \\ y_1 & x_1 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_n & -y_n & 1 & 0 \\ y_n & x_n & 0 & 1 \end{bmatrix}$$

and

$$\boldsymbol{\phi} = [\cos\theta, \sin\theta, dx, dy]^T$$

as before.

The resulting over-determined system can then be solved using standard techniques. As implemented, the system uses an SVD (Singular Value Decomposition)-based solver from the "*GNU Scientific Library*" [45] to find a least-squares solution.

As an intermediate step, the RANSAC (Random Sample Consensus) algorithm [37] can be used to remove outliers, providing a more robust set of features from which to estimate motion. Furthermore, a modified version of RANSAC can be used whereby, instead of the traditional selection of the set of samples with the greatest number of inliers (where inliers are chosen by thresholding an error), the sample set producing the overall lowest error is used (with thresholding still used to remove outliers). The RANSAC algorithm used is shown in Algorithm 3.2.

3.6.2 Quality Control

In spite of the use of a large number of features and the RANSAC algorithm to filter poor data and to provide a more robust motion estimate, there are occasions when the data is simply too corrupted to provide a reliable estimate. This can occur, for example, under particularly poor

Algorithm 3.2: Visual Odometry RANSAC filtering algorithm

Procedure: F_{good} = `RANSACFilter(`F_{raw}`,` N_{trials}`)`

Input: set of N_F raw feature points $F_{raw} = \{(\boldsymbol{p_0}, \boldsymbol{p'_0}), \ldots, (\boldsymbol{p_{N_F-1}}, \boldsymbol{p'_{N_F-1}})\}$

Input: number of trials to perform N_{trials}

Output: set of M_F good feature points $F_{good} = \{(\boldsymbol{p_0}, \boldsymbol{p'_0}), \ldots, (\boldsymbol{p_{M_F-1}}, \boldsymbol{p'_{M_F-1}})\}$ where $M_F \leq N_F$

Data: F ▷ feature set

Data: $\boldsymbol{p}, \boldsymbol{p'}$ ▷ pre-, post-motion position

Data: $f = (\boldsymbol{p}, \boldsymbol{p'})$ ▷ feature

Data: ε, E ▷ error, total error

Data: K ▷ number of inliers

1: $\bar{\varepsilon}_{best} \leftarrow \infty$ ▷ initialise best mean error

2: $F_{good} \leftarrow \{\emptyset\}$ ▷ initialise good feature set

3: **for** N_{trials} **do**

▷ randomly select two features

4: $f_{rand0} \leftarrow RANDOM(F_{raw})$

5: $f_{rand1} \leftarrow RANDOM(F_{raw})$

▷ estimate motion model $\hat{\phi}$ (Eq^n 3.7)

6: $\mathsf{Q}_{\text{trial}} \leftarrow (\boldsymbol{p_{rand0}}, \boldsymbol{p_{rand1}})$ ▷ construct matrix Q (Eq^n 3.6)

7: $\boldsymbol{q'_{trial}} \leftarrow (\boldsymbol{p'_{rand0}}, \boldsymbol{p'_{rand1}})$ ▷ construct vector $\boldsymbol{q'}$ (Eq^n 3.5)

8: $\hat{\boldsymbol{\phi}}_{\boldsymbol{trial}} \leftarrow \mathsf{Q}_{\text{trial}}{}^{-1} \boldsymbol{q'_{trial}}$ ▷ solve Eq^n 3.4 for vector $\hat{\boldsymbol{\phi}}$

9: $K_{trial} \leftarrow 0$ ▷ initialise number of inliers

10: $F_{trial} \leftarrow \{\emptyset\}$ ▷ initialise feature set

11: $E_{trial} \leftarrow 0$ ▷ initialise total error

12: $\hat{\Phi}_{\text{trial}} \leftarrow \hat{\boldsymbol{\phi}}_{\boldsymbol{trial}}$ ▷ construct $\hat{\Phi}$ (Eq^n 3.3)

▷ test each feature in F_{raw} against $\hat{\boldsymbol{\phi}}_{\boldsymbol{trial}}$

13: **for** $i \leftarrow 0$ **to** $N_F - 1$ **do**

14: $(\boldsymbol{p_i}, \boldsymbol{p'_i}) \leftarrow F_{raw}[i]$

15: $\hat{\boldsymbol{p}}'_{\boldsymbol{i}} \leftarrow \hat{\Phi}_{\text{trial}} \boldsymbol{p_i}$ ▷ predict $\hat{\boldsymbol{p}}'_{\boldsymbol{i}}$ using $\hat{\Phi}$ and $\boldsymbol{p_i}$ (Eq^n 3.2)

16: $\varepsilon_i \leftarrow |\boldsymbol{p'_i} - \hat{\boldsymbol{p}}'_{\boldsymbol{i}}|$ ▷ determine error

17: **if** $\varepsilon_i < \varepsilon_{max}$ **then** ▷ compare error to threshold

18: $K_{trial} \leftarrow K_{trial} + 1$ ▷ increment number of inliers

19: $F_{trial} \leftarrow F_{trial} \cup f_i$ ▷ add feature to set of good features

20: $E_{trial} \leftarrow E_{trial} + \varepsilon_i$ ▷ update total error

21: **end**

22: **end**

23: **continued ...**

Algorithm 3.2: continued

```
... continued
    ▷ only consider this trial if there are at least two good features
    if K_trial ≥ 2 then
        ε̄_trial ← E_trial / K_trial                          ▷ calculate mean error
        if ε̄_trial < ε̄_best then                  ▷ compare error against best error
            ε̄_best ← ε̄_trial                                  ▷ store new best error
            F_good ← F_trial                          ▷ store new best feature set
        end
    end
end
  ▷ F_good now contains the most consistent set of features
```

lighting conditions, over surfaces with uniform appearance (such as concrete or clean sand) or when the robot is moving faster than can be estimated, all of which produce erroneous and inconsistent feature displacement estimates. To alleviate this, we introduce a simple measure of motion quality. We make the assumption that a valid and reliable motion estimate is one that results from a consistent data set, or in other words, one that is produced and hence supported by a high proportion of data. The measure of quality used is simply the proportion of features used to produce the motion estimate. We then threshold the quality of motion estimates and only consider those with a high enough quality to be valid. The threshold used, determined empirically, is 0.3. Motion estimates of poor quality – those below the threshold – are flagged as being invalid and are ignored.

3.6.3 Parameter Selection

There are three parameters that control the visual odometry system: the search window size, the number of features to use and the feature (or template) size. While these are ultimately independent in terms of the ability to choose each of them separately, their influence on performance is tightly coupled.

As a first comment it should be noted that processing rate, the rate at which motion can be resolved, has a strong bearing on the maximum measurable robot speeds. For a given processing rate, an increase in robot speed means greater displacements of individual features between frames. The first limit of usability is reached when the displacement of a feature is so great that the feature no longer falls within the search window in the following image. At this stage, the search window size can potentially be increased to allow the displacement to be detected. The second limit occurs when a feature no longer falls within the bounds of the entire image. The parameters may be able to be adjusted to increase the processing rate and hence reduce the displacement of the feature between frames. The extreme limit occurs when the feature no longer falls within the bounds of the entire image *and* the system is running at or above the rate at which frames can be captured (25 frames-per-second in a PAL video system). At this point, there is no change of parameter that will allow the feature to be recovered. However, changes in hardware or physical setup may be made to accommodate such speeds as discussed in Section 3.6.4.

Search Window

The search window size has a clear and direct relationship to the range of motion that is able to be estimated. A larger search area means greater individual feature displacements can be recovered resulting in a larger range of overall translations and rotations being measurable. For a given processing rate, this means an increase in the speeds of the robot that can be measured, both translational and rotational. Increasing the size of the search window, though, means more matching operations are required for each feature, so the processing rate is reduced, resulting in a reduction in the robot speeds measurable. Choosing the search window size is a process of balancing the range of measurement against processing speed.

Number of Features

An increase in the number of features used results in a more accurate estimation of motion as the effects of individual errors are reduced. Especially when there are many erroneous matches (caused, for example, by unmoving shadows cast by the robot itself), using more features can mean having a higher proportion of good versus bad matched features. However, for a given template size and window size, an increase in the number of features used also results in a decrease in speed. The selection of the number of features must balance desired accuracy with processing rate.

Template Size

Up to a point, increasing the template size increases the reliability of the matching process, as each feature is now represented by more information and hence is closer to being unique and therefore being able to be uniquely matched. Beyond a certain size, however, the apparent distortion of a feature under rotation becomes too great, meaning the range of angles-of-rotation able to be reliably detected is reduced. Also, as expected, an increase in feature size requires greater processing for each feature matching operation and hence reduces speed. In selecting a template size, reliability must be balanced against the expected range of rotation and the processing rate achievable.

3.6.4 Hardware Considerations

There are two primary considerations to be made with regards to the hardware used in the visual odometry system. Firstly, there is the camera and more importantly the choice of lens. Secondly, there is the placement and mounting of the camera on the robot platform.

In order to translate a set of feature positions expressed in image coordinates (that is, pixels) into world-coordinates (millimetres), we make use of both the camera's intrinsic parameters (most importantly the camera's focal length) and also its extrinsic parameters, as we are assuming all points in an image lie on a ground plane predetermined though calibration. For a point in the camera's reference frame $\boldsymbol{p_c} = [x_c, y_c, z_c]^T$, the corresponding image point $\boldsymbol{p_i} = [x_i, y_i]^T$ is given by the following basic equations of projection:

$$x_i = \frac{f}{z_c} x_c \tag{3.17}$$

$$y_i = \frac{f}{z_c} y_c \tag{3.18}$$

where f is the camera's focal length, expressed in image units (pixels) and obtained via intrinsic calibration.

If we assume that all image points lie on the same ground plane and, through calibration, the ground plane is parallel to the image plane (and so all points share the same z_c), we have the following:

$$x_c = \frac{z_g}{f} x_i \tag{3.19}$$

$$y_c = \frac{z_g}{f} y_i \tag{3.20}$$

where z_g is the distance between the camera origin and the ground plane, expressed in millimetres and obtained through extrinsic calibration. The factor $\frac{z_g}{f}$ can then be seen as a gain factor that determines the pixel-to-millimetre translation and hence the range (for a given search window) and resolution of measurable motion. With more than one camera, it is not critical to match focal lengths and mounting heights, though for consistency the gain factor $\frac{z_g}{f}$ for each camera should be kept similar. Within the constraints of available hardware and feasible mounting positions, the focal length and camera height can be adjusted to provide a useful range and resolution of motion.

In the setup used, the front camera focal length is 400 pixels (using images of 320×240 pixels), and the camera height is 1010 millimetres, giving approximately 2.5 millimetres-per-pixel, while the rear camera focal length is 270 pixels (using images of 320×240 pixels), and the camera height is 720 millimetres, giving approximately 2.6 millimetres-per-pixel. While not identical, the two cameras have gain factors that are very similar.

Using the base parameters given in Table 3.1 and a gain factor of 2.5 millimetres-per-pixel (the lower of the two), the maximum measurable displacement of a feature is then approximately 37 millimetres horizontally or vertically (with respect to the image) and approximately 53 millimetres diagonally. The maximum straight-line measurable speed of the robot, operating at the stated 15 frames-per-second, is approximately 0.55 metres-per-second, a slow walking pace.

As a final note on the effect of camera placement, we refer to earlier discussion of the benefits of using multiple cameras. Using a pair of cameras, located on opposite sides of the likely centre of rotation, we are much better able to estimate rotations when a single camera cannot effectively be mounted near the centre of rotation. The further each camera is from the centre of rotation, the higher the resolution of rotation estimation. Quite simply, a rotation will produce a tangential translation that is proportional to the radial distance from the centre of rotation. As such, for a fixed search window, a camera mounted close to the centre of rotation will have a large range of rotation estimation, but low resolution, whereas a camera mounted far from the centre will have a low range but high resolution. A compromise must be found between the desired range and the desired resolution (and hence accuracy). Furthermore, the size and shape of the robot must be considered; cameras mounted at the end of long beams on the sides of the robot will provide great rotational resolution, but will significantly reduce manoeuvrability.

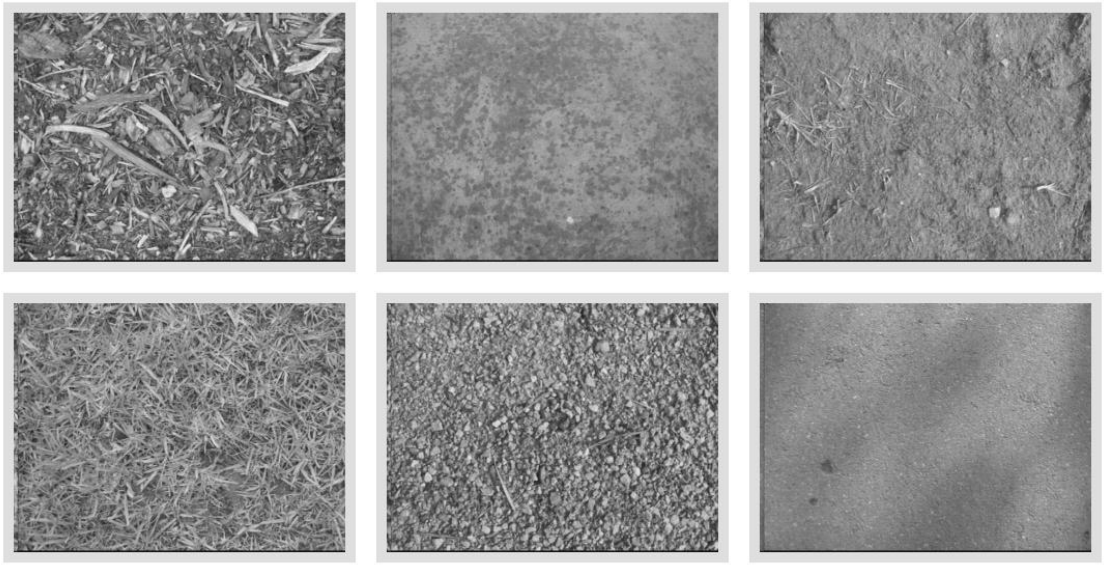


Figure 3.4: A sample of the test images used

3.7 Experimental Results

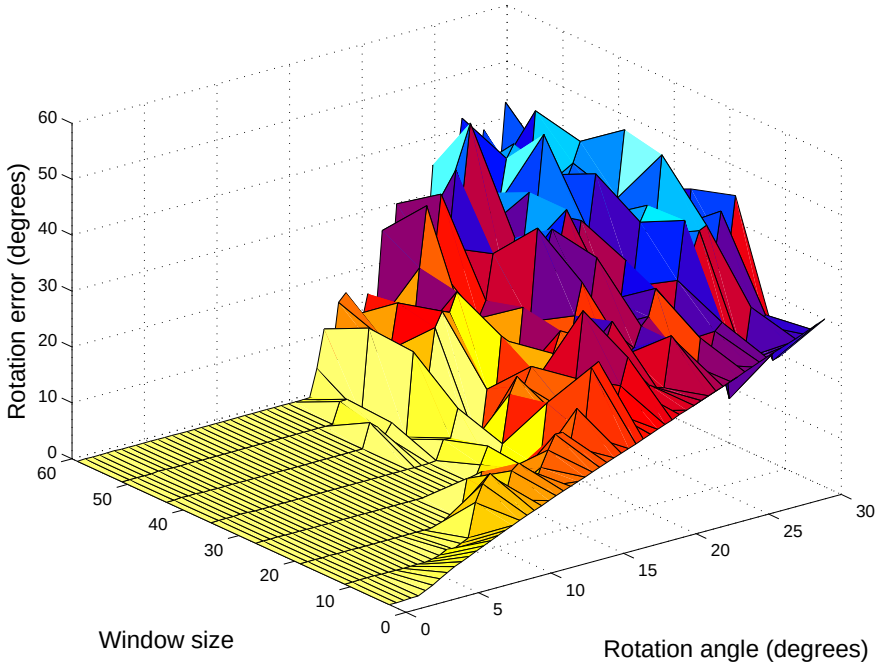
3.7.1 Simulation of Motion

Simulations have been performed to test the effect of parameter variation in the implemented algorithms under controlled conditions. Testing consisted of taking a captured image, performing a translation/rotation transformation on the image and using the original and transformed images as inputs to the odometry estimation routine. The source image is captured at higher-than-normal dimensions before being rotated and translated by known amounts. Finally, the source and transformed images are both cropped down to standard dimensions. These two steps - large-size capture and final cropping - are done to ensure edge-effect distortions that result from the transformation do not appear in the transformed image being used.

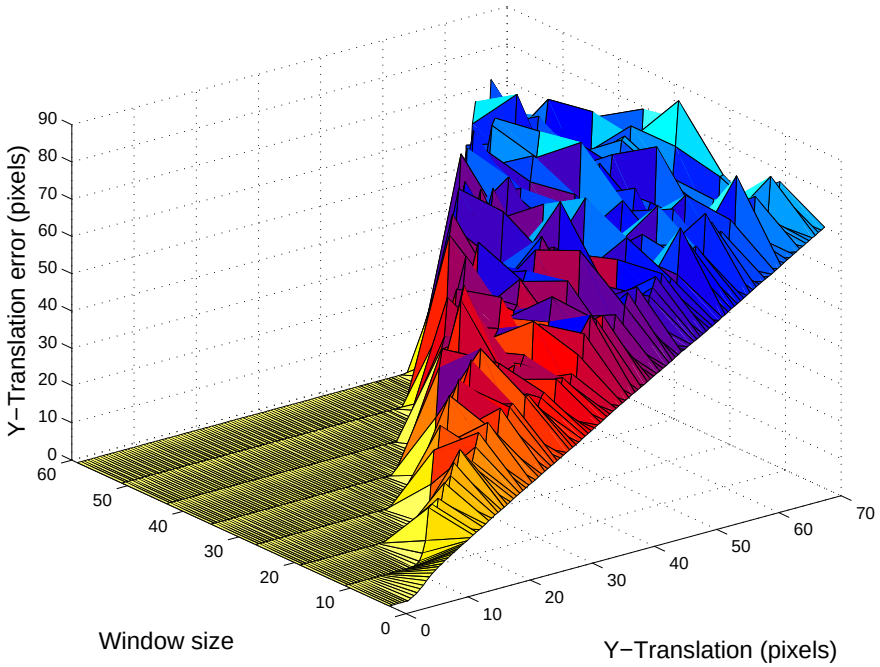
A sequence of fourteen input images were used, consisting of images taken of a variety of ground types, including concrete, tarmac, mud/dirt, gravel, grass and combinations. A selection of the images used is presented in Figure 3.4. Transformations were applied to all images and the results shown here are the averages over the fourteen images.

In each test, two of the three paramters above were held constant (to the values shown in Table 3.1) while the third (the control) was varied. For each control parameter value, the rotation angle and a translation distance (in one direction) was swept through a range of values, and the error (difference between estimated motion and applied transformation) was measured.

An increase of the window size, as shown in Figure 3.5 effectively increases the range of motion (either rotation or translation) able to be measured, although the effect is much greater on measurable translation than rotation. The number of features, shown in Figure 3.6, is not seen to have a significant effect on errors. This is primarily because there is no noise present in the simulation meaning all template matches are likely to be very good. In Figure 3.7, a

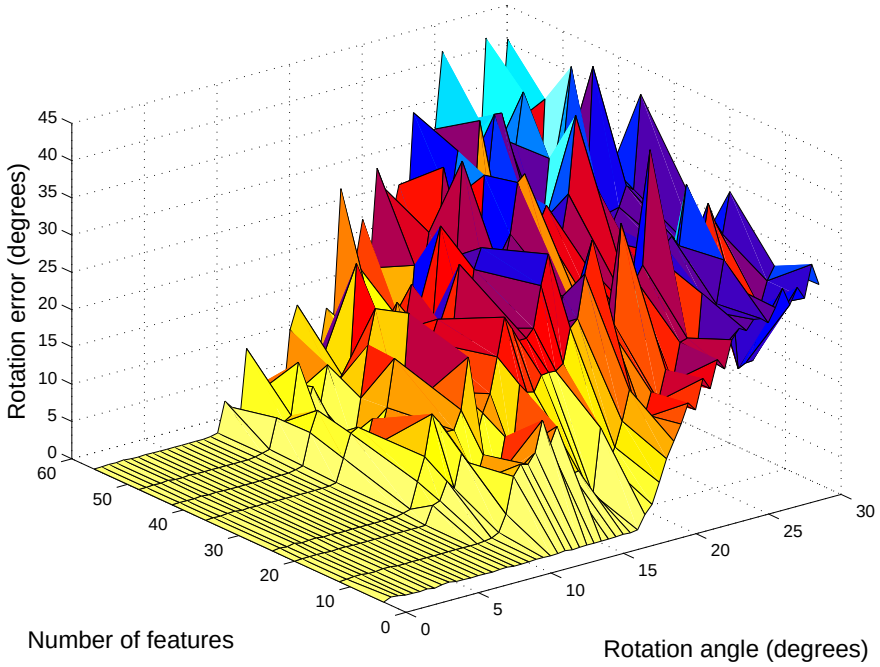


(a) Control: window size, Variable: rotation angle

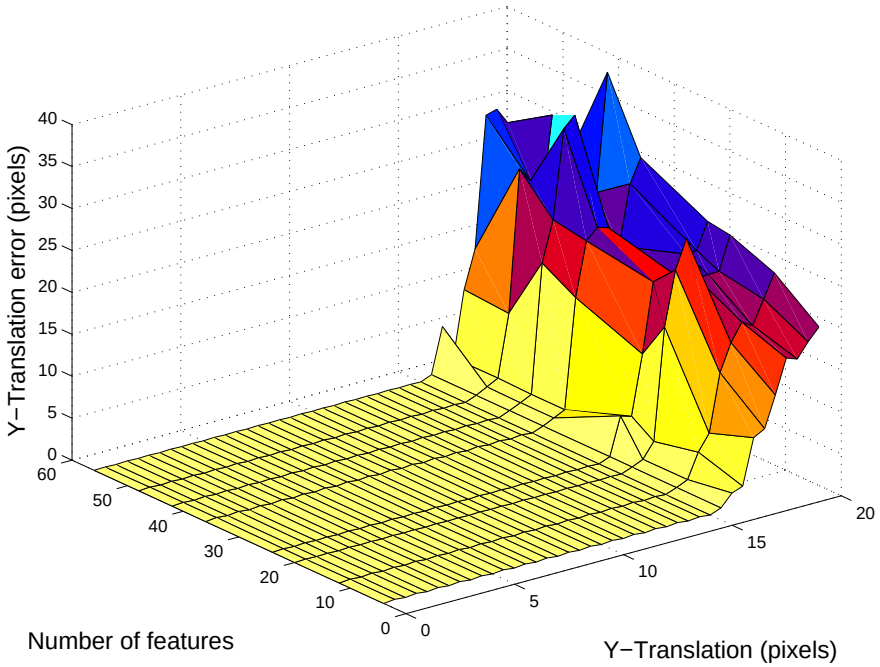


(b) Control: window size, Variable: translation

Figure 3.5: The effect of varying the search window

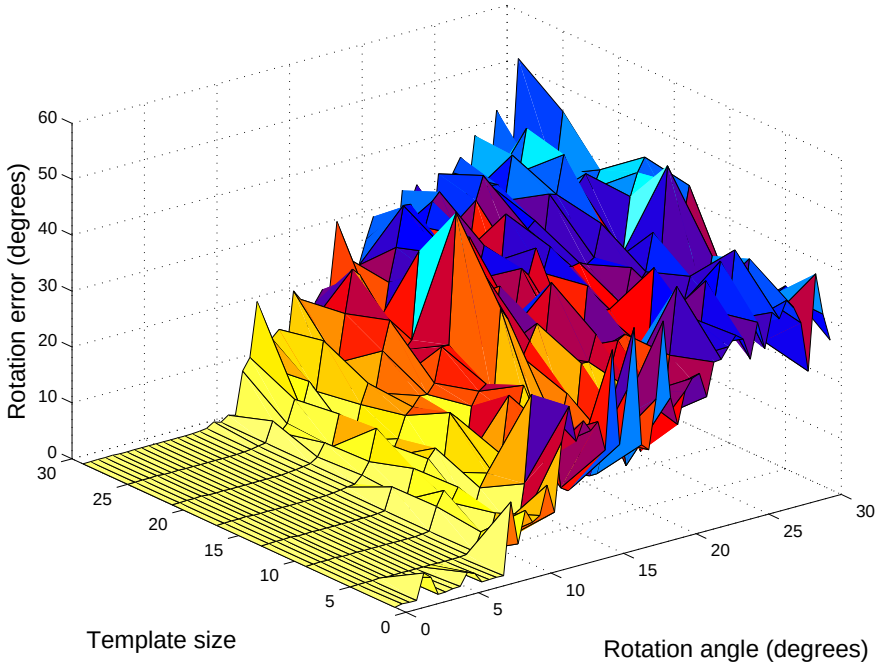


(a) Control: number of features, Variable: rotation angle

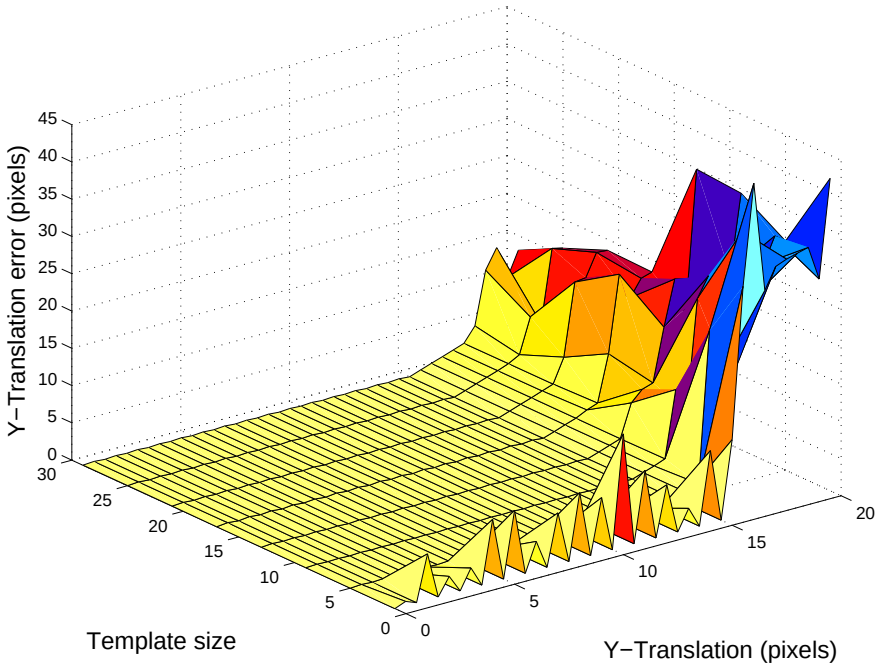


(b) Control: number of features, Variable: translation

Figure 3.6: The effect of varying the number of features



(a) Control: template size, Variable: rotation angle



(b) Control: template size, Variable: translation

Figure 3.7: The effect of varying the template size

Parameter	Value used
Search window	
C_ω	15
R_ω	15
overall	31×31
Template	
C_τ	7
R_τ	7
overall	15×15
Number of features	48

Table 3.1: Base parameters for visual odometry simulation

similar result is seen. Above a minimum template size (approximately five pixels), there is no effect of template size on error. Again, because there is no noise, template matches are good, even with small templates (above a minimum).

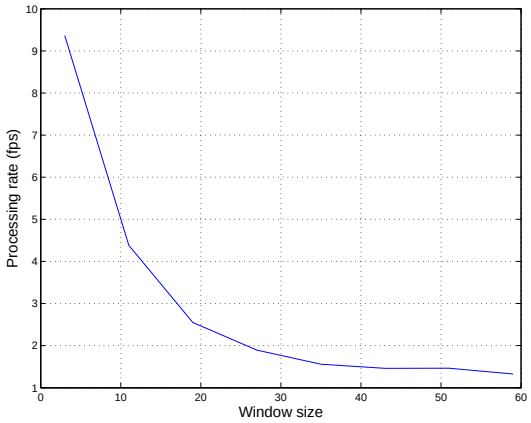
In addition to measuring the error, the approximate processing rate for each control value was measured. Graphs of control value versus processing rate are shown in Figure 3.8. The actual processing rates shown should be taken as a rough guide to performance only; what is important is the way the rate changes as the parameters are changes. As clearly shown, increasing any of the parameters serves to reduce the rate in an approximately inversely-proportional manner.

3.7.2 Physical Experimentation

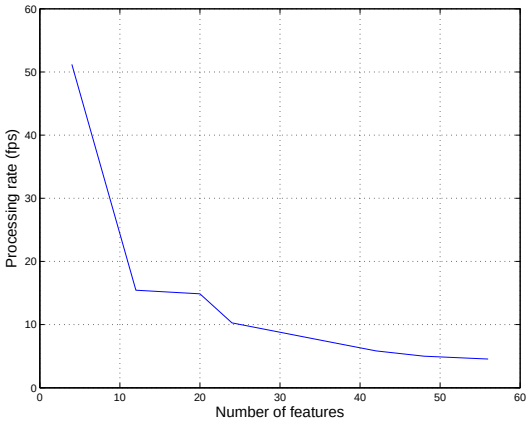
This section describes a series of physical experiments, in which a measured course was created on a grassy field and the robot manually driven around it. Image sequences were captured while driving the robot and processing was performed off-line. This was done for two reasons: firstly, it allowed parameters to be changed to see their effect and secondly – and unfortunately – because it was found that the online processing speed (in the order of eight frames-per-second when using front and rear images with the desired parameters) was not adequate to allow the robot to travel at a reasonable speed. The parameters used for the experiments, determined empirically, are shown in Table 3.2.

The first set of results use a large rectangular course with the robot driven around in a simple loop. The dimensions of the rectangle were 15 metres by 10 metres, measured using a 30 metre measuring tape and marked (every 5 metres) with pegs in the ground. Starting at one corner, the robot was driven anti-clockwise around the loop twice. The course and path are shown in Figure 3.9.

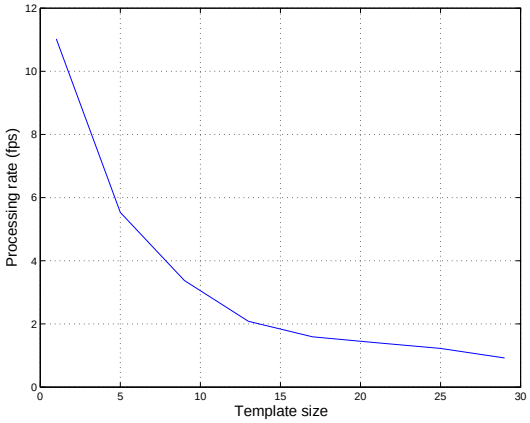
In the results shown in Figures 3.10, 3.11, 3.12 and 3.13, a comparison is made between the use of all estimates and use of only quality-filtered estimates, as per the use of motion estimate quality thresholding, described in Section 3.6.2. As well as this, a comparison is shown between the use of both front and rear cameras, front camera only, and rear camera only.



(a) Control: window size



(b) Control: number of features



(c) Control: template size

Figure 3.8: The effect on processing rate

Parameter	Value used
Search window	
C_ω	20
R_ω	20
overall	41×41
Template	
C_τ	7
R_τ	7
overall	15×15
Number of features	48 per image (96 in total)

Table 3.2: Visual odometry parameters for physical experiments

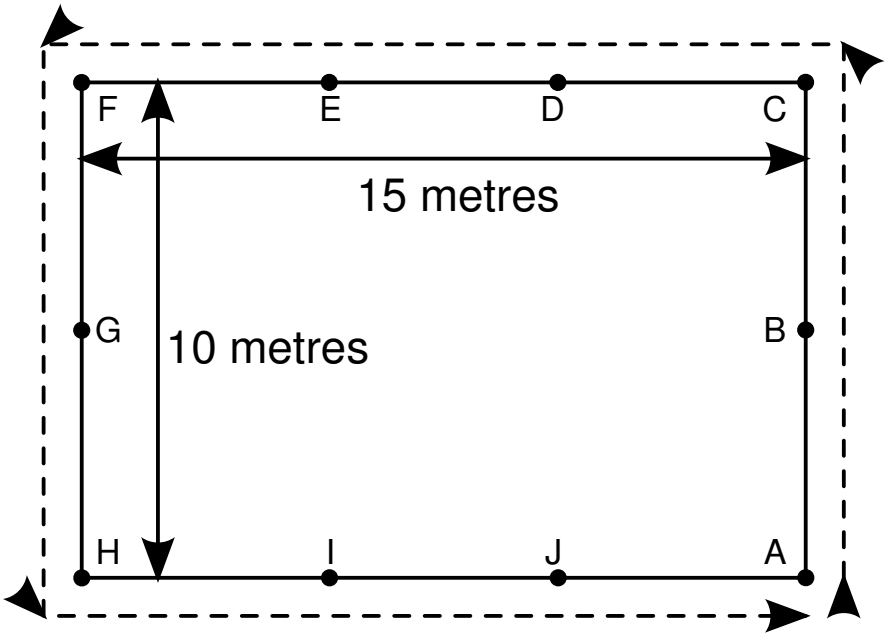


Figure 3.9: Path One

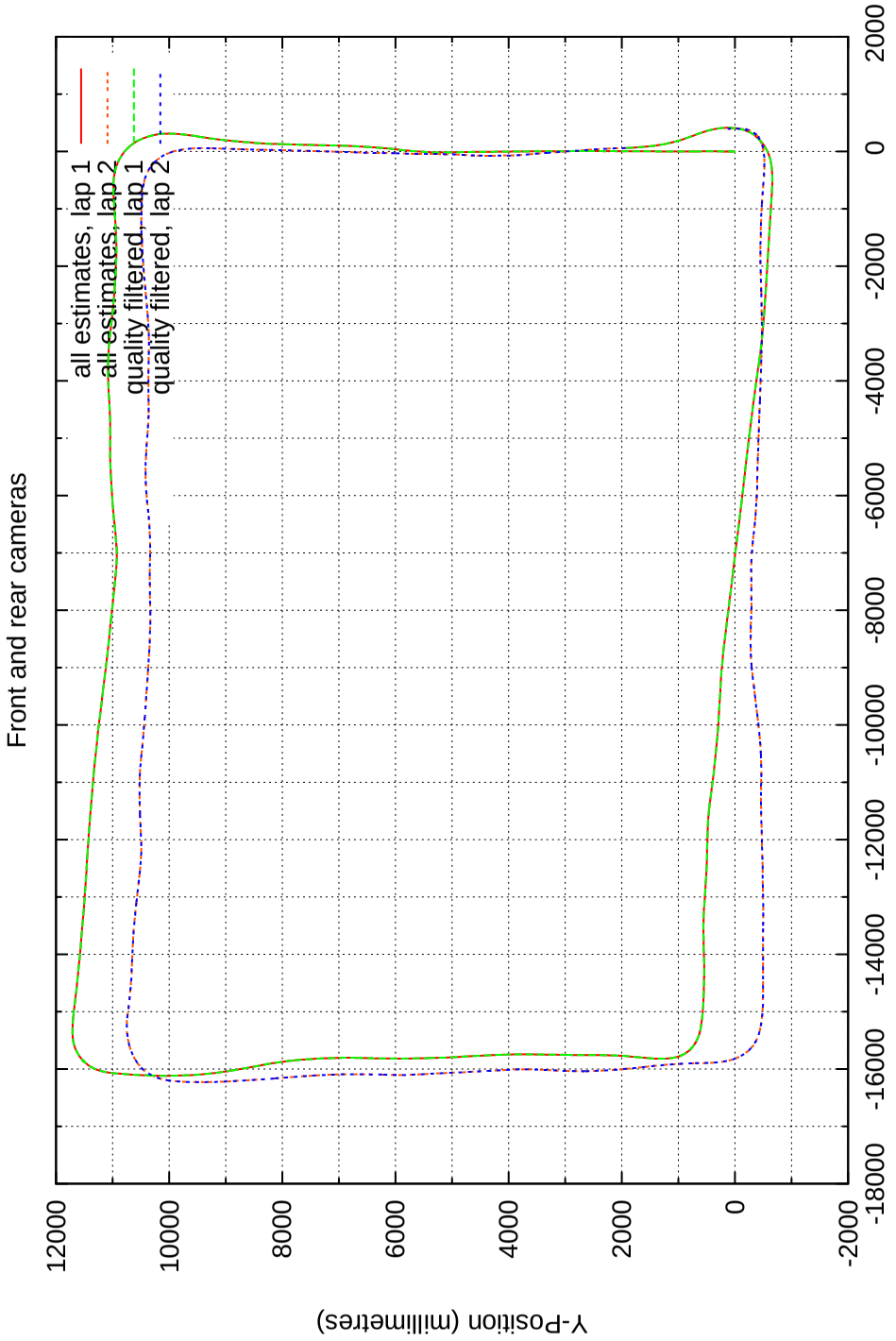


Figure 3.10: Path One: Front and Rear cameras operational

One of the first things to note with these results (primarily those shown in Figure 3.10) are the lengths of the path segments estimated. Taking into account that the robot travelled outside the bounds of the rectangular course, the results are very good. The length of edges $\overrightarrow{AC}$ and $\overrightarrow{FH}$ (referring to the labelling of locations in Figure 3.9) are both approximately 12 metres and edges $\overrightarrow{CF}$ and $\overrightarrow{HA}$ are approximately 16 metres, consistent with the robot travelling outside around a 10×15 metre course with its centre approximately a metre from the course edge (corresponding to a buffer of approximately 0.75 metres between robot edge and course edge). From this we can conclude that – with appropriately accurate calibration – the assumption of a locally flat ground is valid, as an error in this assumption would lead to incorrect scaling of image feature displacements to ground feature displacements.

A second positive result (again taken from Figure 3.10) is shown where, after the first complete loop is performed, the estimated path very closely follows that of the initial traversal of the now-repeated edge. This is a clear indication of the system's ability to "*close the loop*", that is, correctly indicate that two otherwise independent locations (as the system does no global comparisons) are actually the same location. Alternatively stated, this is an indication of the system's low cumulative – or drift – error. After travelling 60 metres (one complete loop plus a repeated 10 metre segment from A to C), there is a positional error in the order of 1 metre in the Y direction, shown by the misalignment of edges $\overrightarrow{CF}$ and $\overrightarrow{HA}$ on the second lap. This corresponds to a positional error of approximately 1.5 percent. The error in rotation (the component of motion that would produce the worst results if erroneous) is virtually zero, shown by the directional alignment of all edges.

Looking at Figures 3.11 and 3.12 where only front *or* rear cameras were used, we see a reduction in the accuracy of estimated motion, primarily the estimation of rotation. When using the front camera, there is a tendency to underestimate rotation, leading to a positional error in the order of 2 metres after one loop and 6 metres after two complete loops. The error in orientation is approximately 15 degrees after one lap and approximately 33 degrees after two laps, calculated as the angle of edge $\overrightarrow{HA}$ against the horizontal. Using the rear camera only, there are again errors in rotation estimation, in this case over-estimations. After one lap there is a positional error of approximately 1.5 metres, while after two laps this has increased to approximately 3 metres. The error in orientation after one lap is approximately 4 degrees, increasing to approximately 18 degrees after two laps. A summary comparison of the results produced using each camera combination is shown in Figure 3.13 which shows clearly that the best results are obtained using both front and rear cameras.

Of final note is the distinct similarity between paths produced using all motion estimates and those produced using only quality-filtered estimates. This is the result of a combination of there being few estimates of quality below the threshold and those estimates of poor quality not being highly erroneous. Of the 8300 data points in each of these sets, the average quality was 0.97 using both cameras, 0.99 using front only and 0.99 using rear only. In all cases there were only four estimates below the quality threshold, each producing a zero-magnitude motion estimate, resulting in no difference between the paths produced using all estimates and those using only filtered estimates.

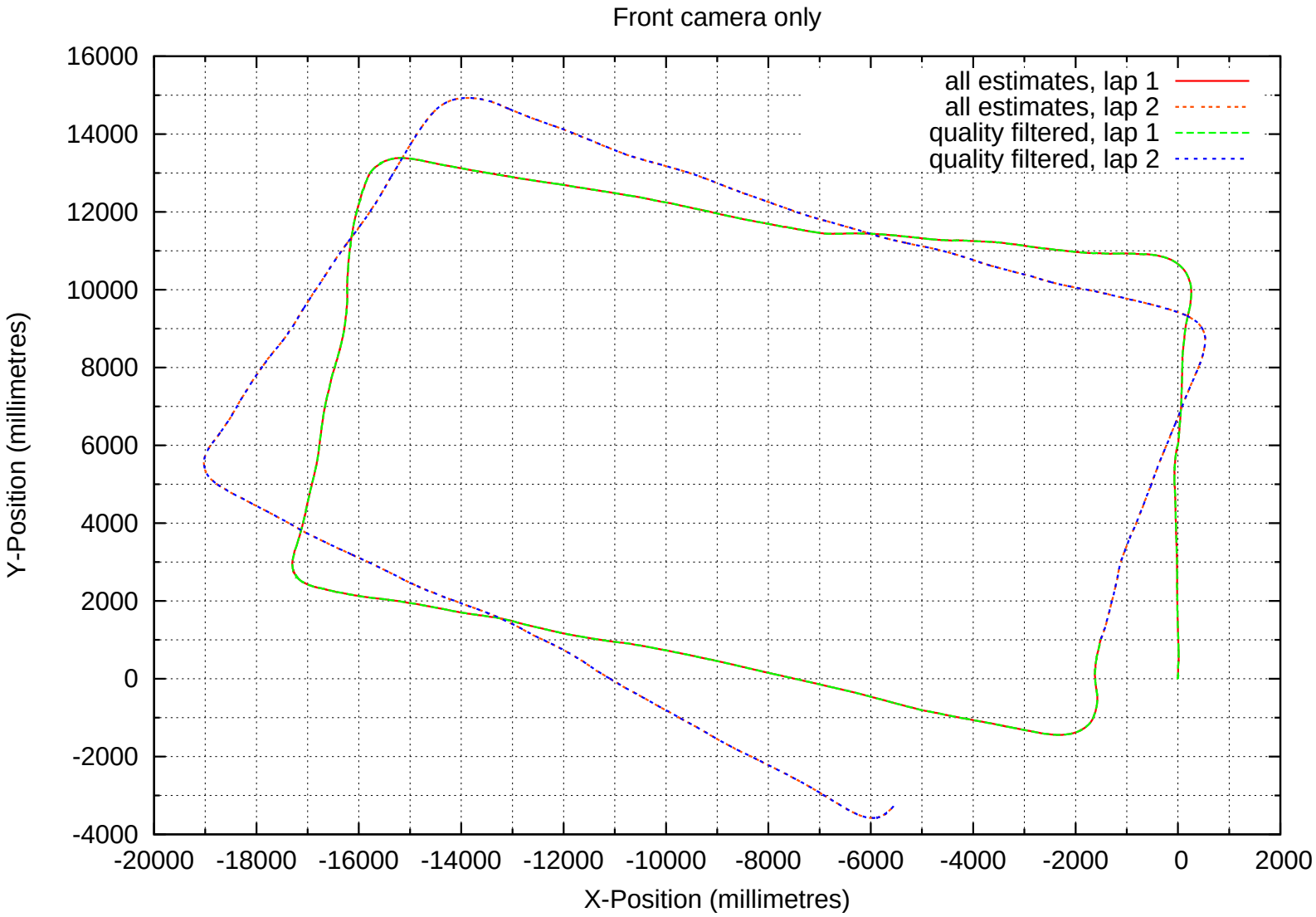


Figure 3.11: Path One: Front camera only

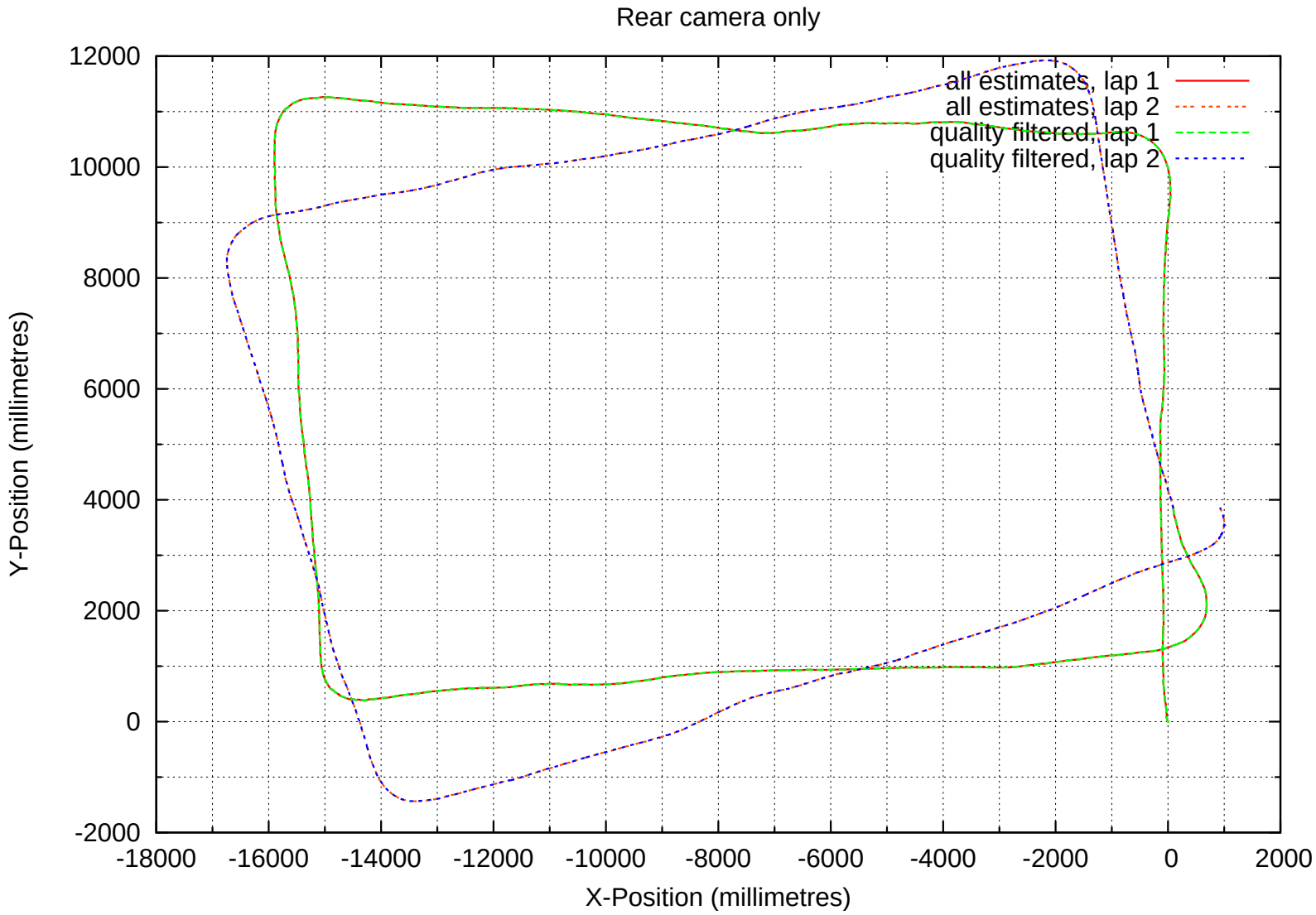


Figure 3.12: Path One: Rear camera only

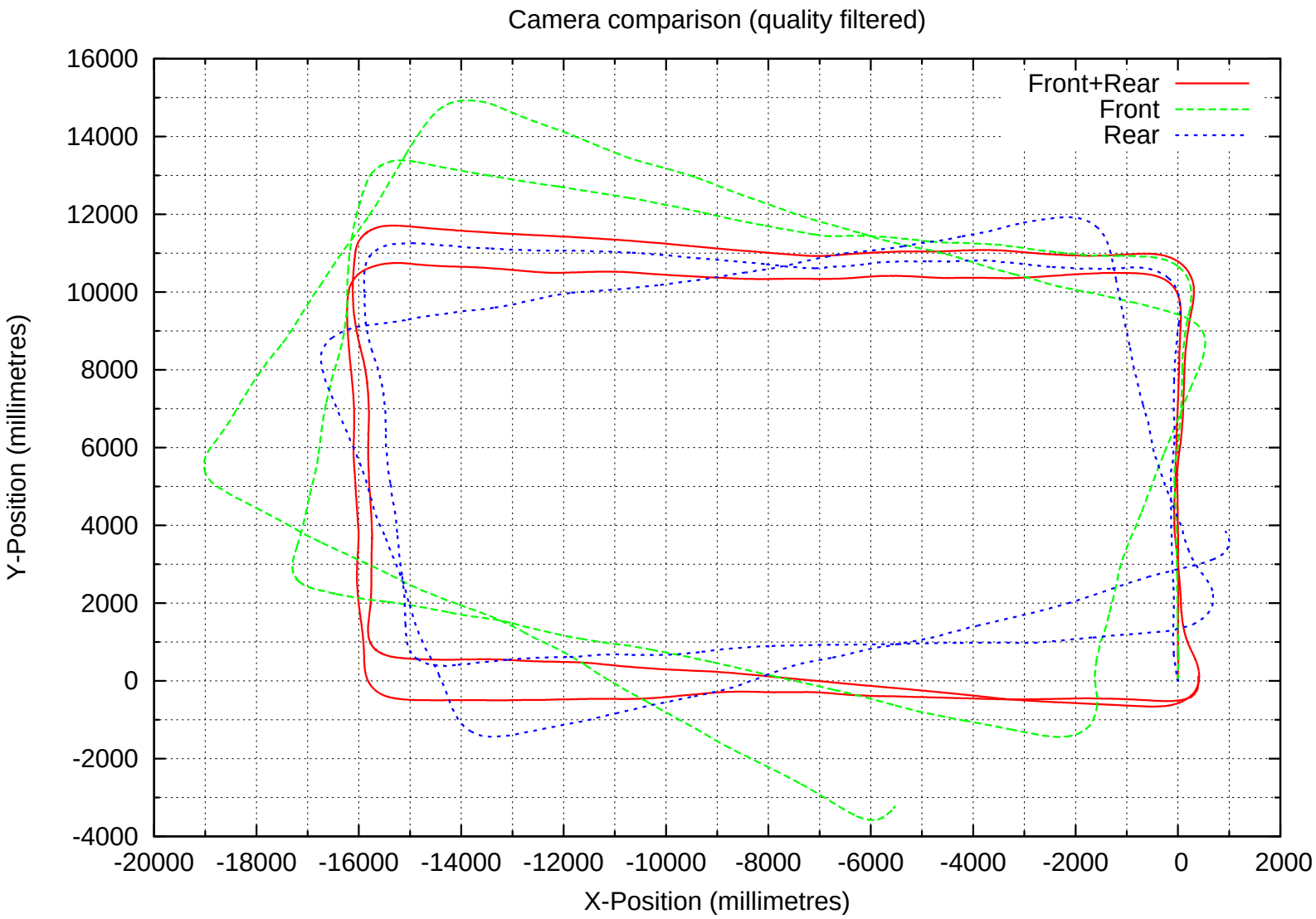


Figure 3.13: Path One: Comparison of camera choices

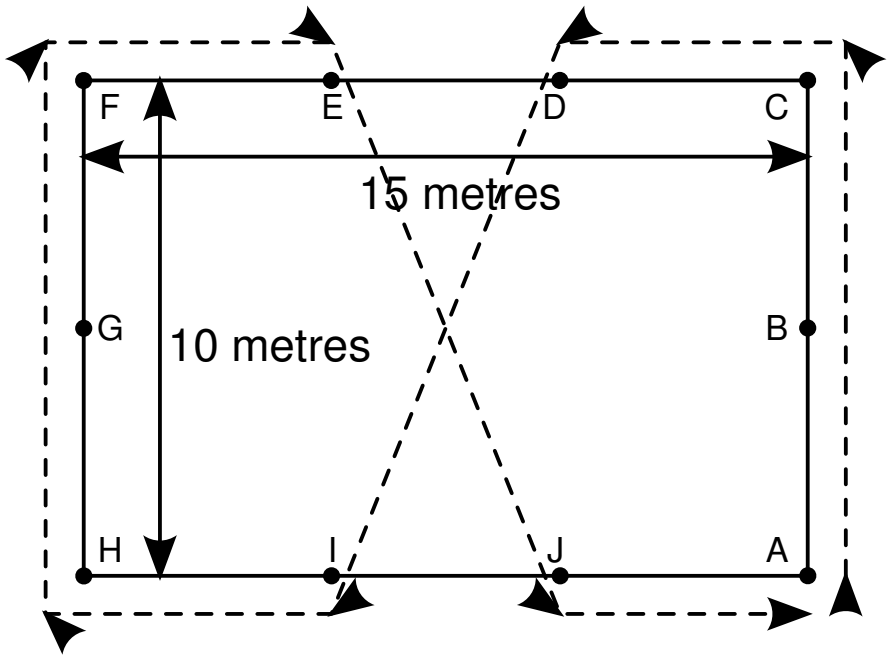


Figure 3.14: Path Two

The second set of results used the same rectangular course, but this time a "*figure-of-eight*" path was taken, as depicted in Figure 3.14. With twice as many corners in this path as the last, it is expected that performance will suffer. Figure 3.15 clearly shows that the path produced is more erroneous than that in the previous experiment. Edges $\overrightarrow{AC}$, $\overrightarrow{CD}$ and $\overrightarrow{DI}$ are quite good with almost no positional error at location I, though an accumulated orientation error means by location H (a distance of approximately 30 metres) there is a positional error in the order of 1 metre in the Y direction, that is, a positional error of approximately 3 percent. Edge $\overrightarrow{HF}$ has an approximately correct length of 11 metres though there is an orientation error of approximately 6 degrees, leading to a positional error at F of approximately 1.5 metres. These errors continue to accumulate, resulting in a positional error of approximately 3.5 metres after one lap (an error of approximately 5.6 percent). The path produced using all motion estimates is even worse, primarily due to two highly erroneous estimates, the first half-way along edge $\overrightarrow{FE}$ on the first lap and the second near the end of edge $\overrightarrow{DI}$ on the second lap. In both cases the erroneous estimate can be seen as a sharp discontinuity in an otherwise smooth path. In total, 16 estimates were of below-threshold quality and within this group the mean motion estimate magnitude was 0.047 metres with a maximum of 0.311 metres. Results are similar when using only one camera, with the front camera producing much worse results (much greater orientation error), and the rear camera producing a path with a similar level of error.

These results do show clearly the benefit of filtering motion estimates on the basis of quality, highlighted in the significant improvement of the path produced when using both cameras. What is most clear, however, is that the system is still susceptible to accumulated errors and, as anticipated, it is errors in the estimated rotation (leading to orientation drift) that are most detrimental to accuracy.

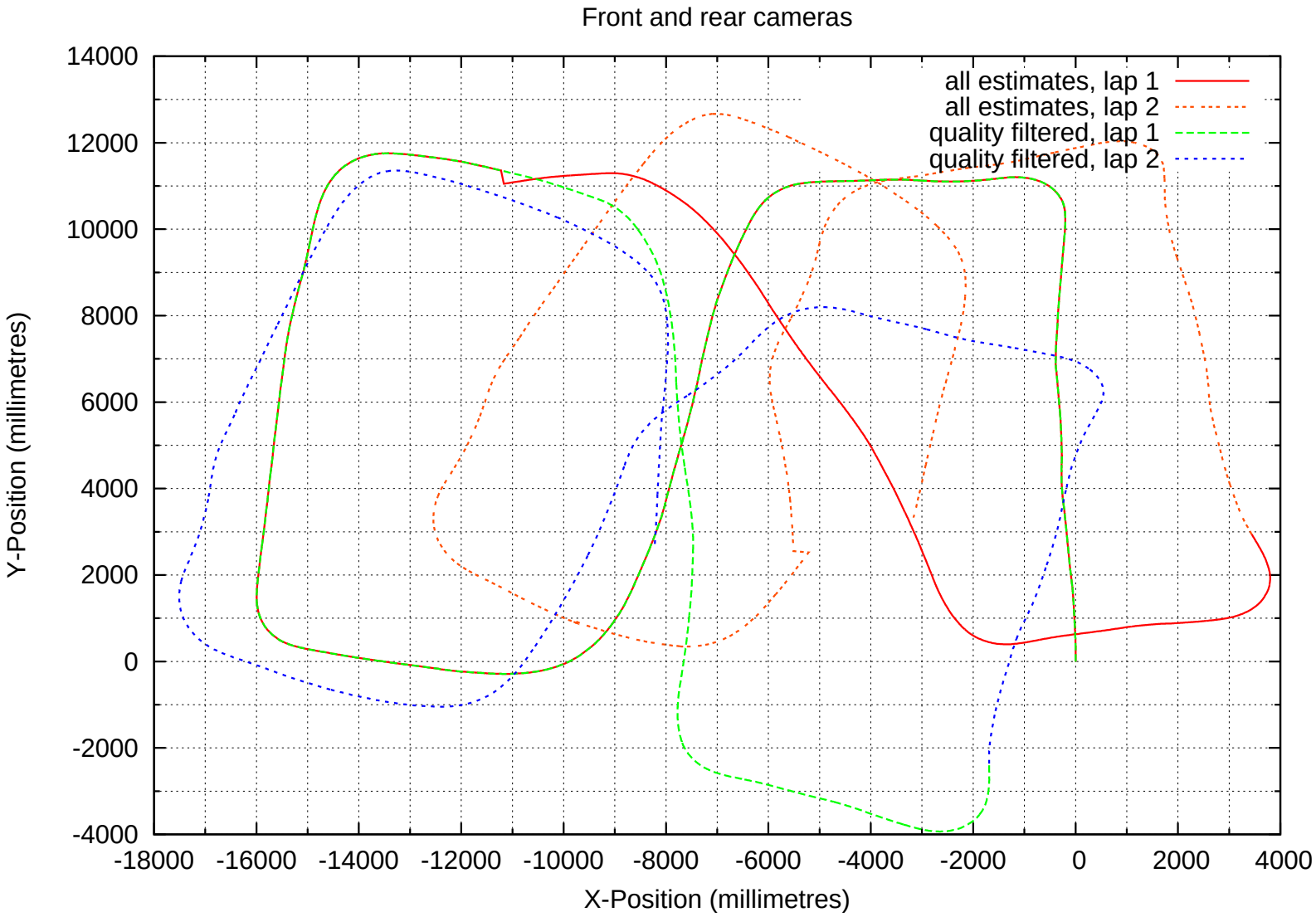


Figure 3.15: Path Two: Front and Rear cameras operational

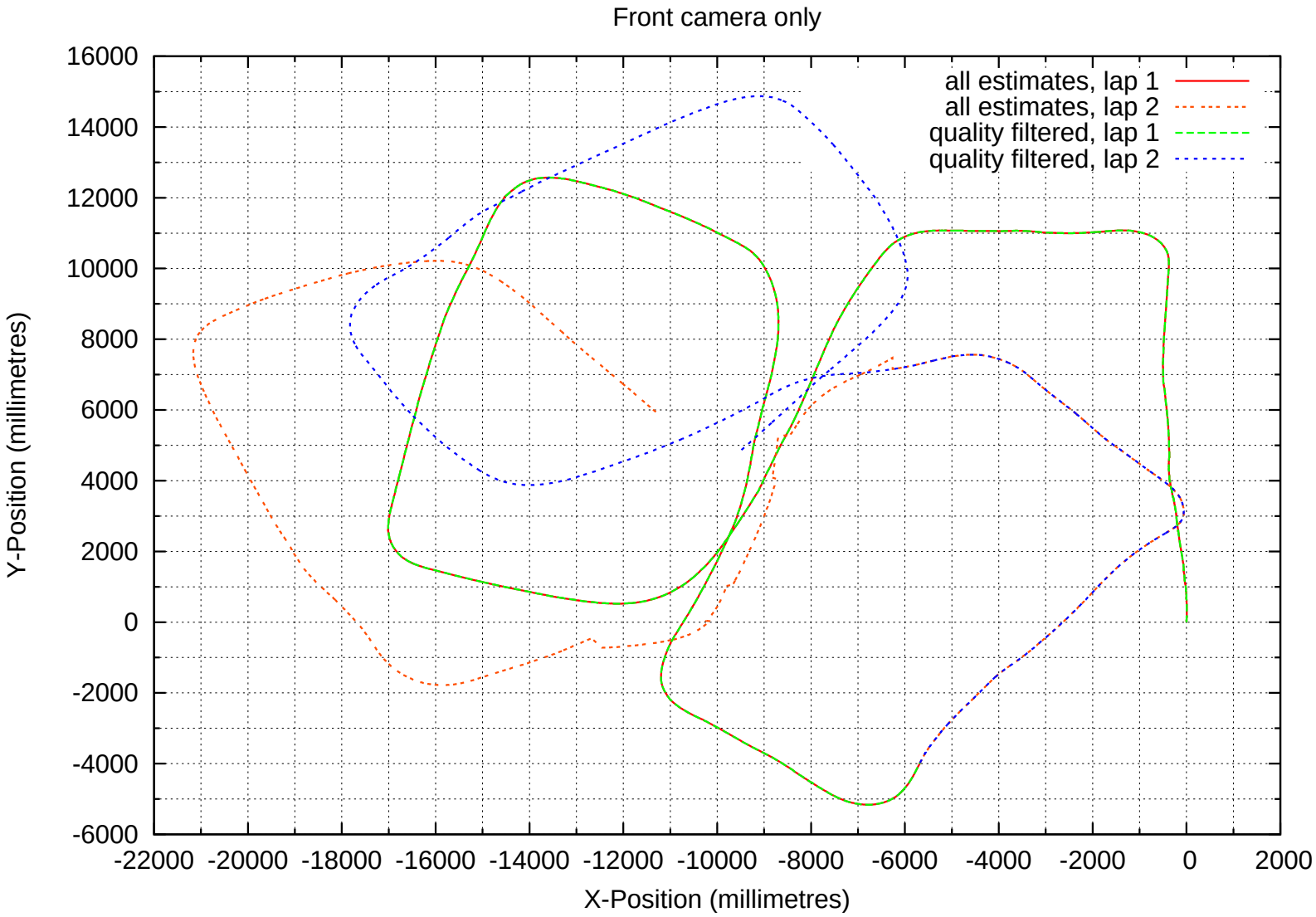


Figure 3.16: Path Two: Front camera only

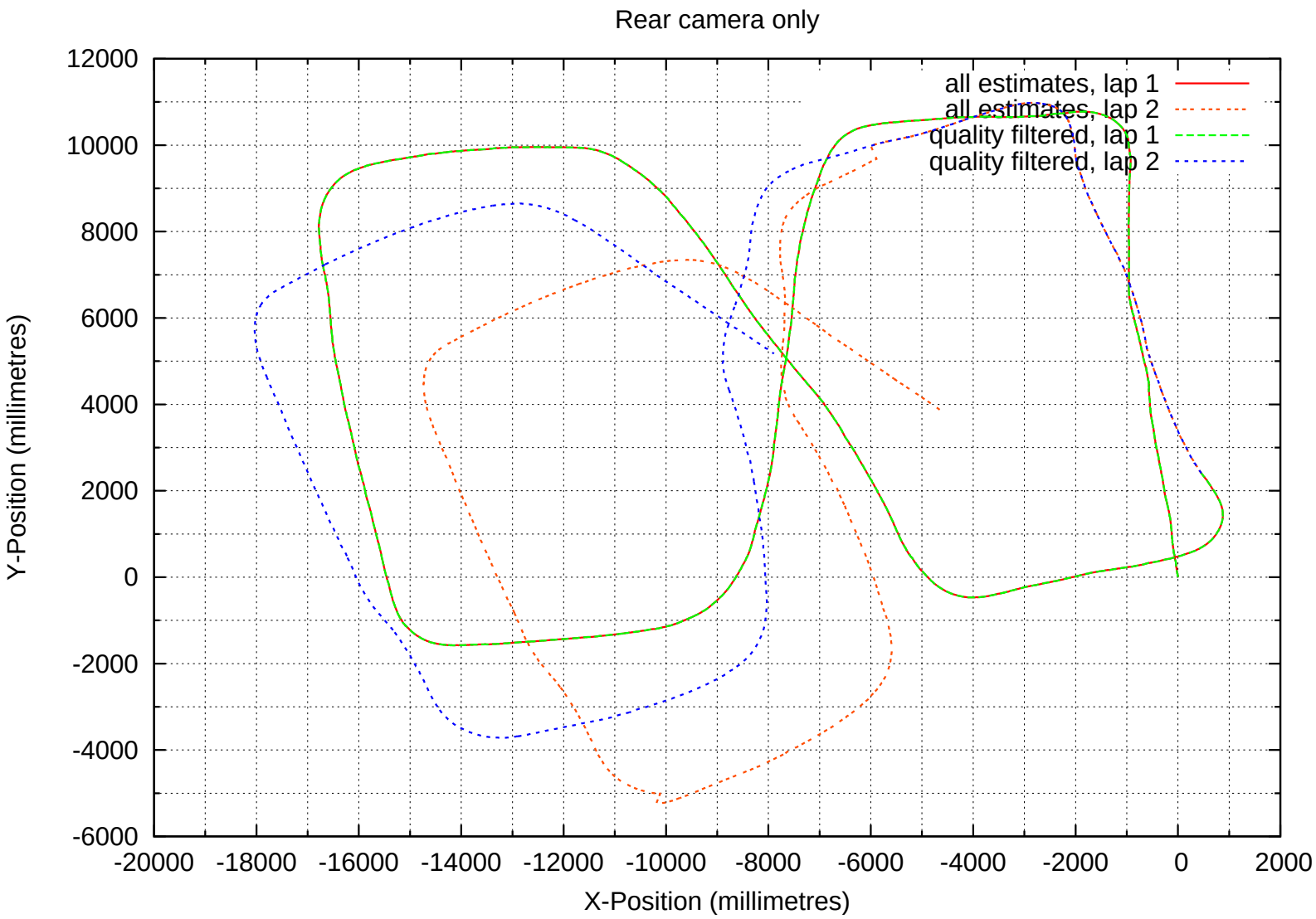


Figure 3.17: Path Two: Rear camera only

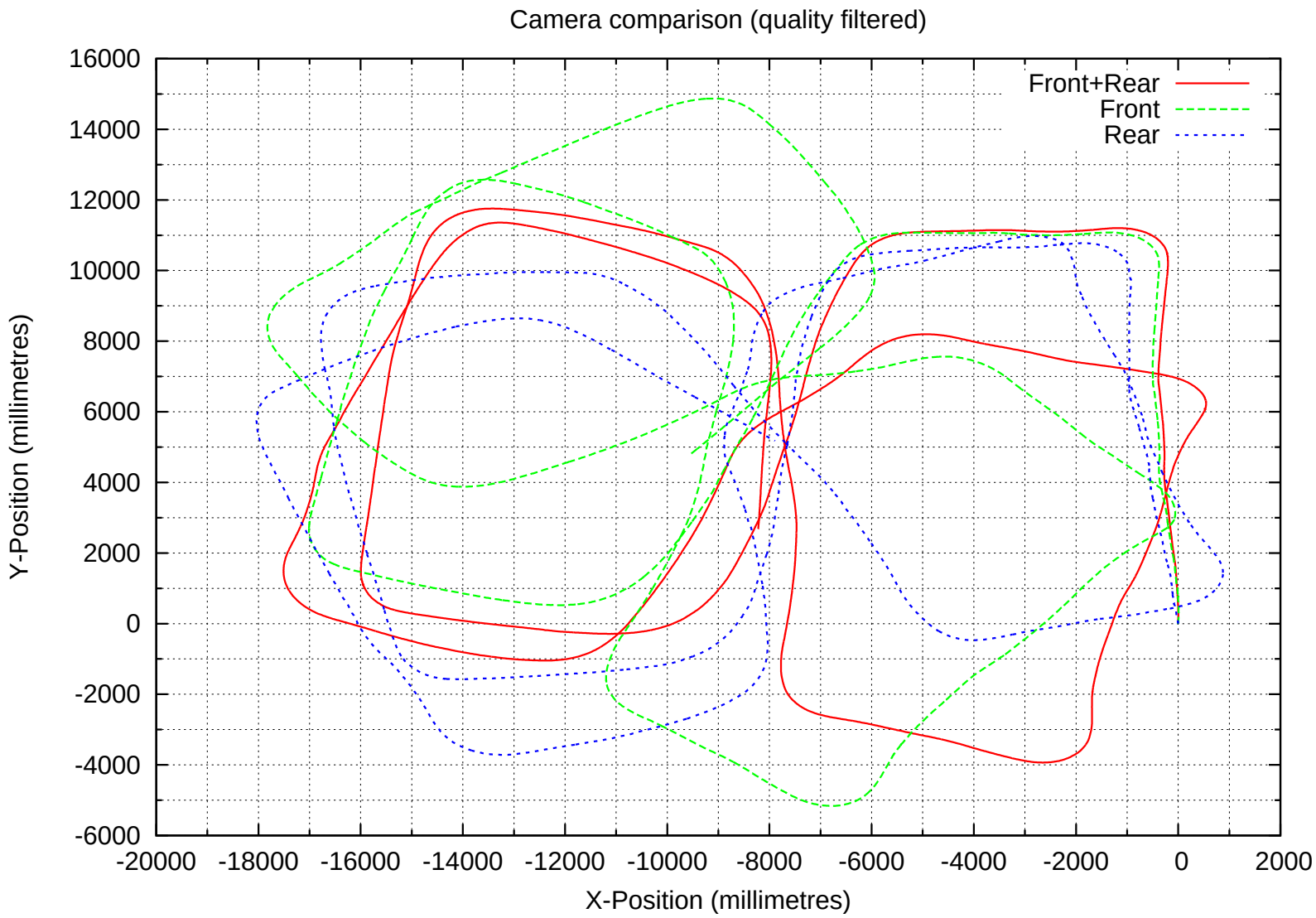


Figure 3.18: Path Two: Comparison of camera options

As a final word, we refer back to the intended use of this system. The visual odometry system presented in this chapter is intended to be used for relative localisation, primarily for motion-control and to control the execution of relatively small-scale path traversal. Accordingly, it need not be impervious to accumulated error. A positional error in the order of five percent after a fifty metre journey – an error of two-and-a-half metres – is considered more than adequate for such use. The results presented above show that this level of accuracy is achievable and as shown in the first experiment able to be exceeded.

3.8 Conclusion

In this chapter, we first highlighted one of the key problems this thesis aims to solve, namely, how can a robot's motion be estimated or alternatively, how can we perform relative localisation, both reliably and accurately as the robot moves through unpredictable terrain. One of the key aspects is that it is very common in largely natural outdoor environments for a robot to slip in any direction as it attempts to move (or as it attempts to remain stationary), meaning traditional solutions involving wheel encoders (even those mounted separately to the driven wheels) are prone to errors and discrepancies between the measured motion and the actual motion.

We proposed a novel solution using passive vision and optical flow techniques to observe the ground near the robot and, from a sequence of images, estimate the motion of the ground and hence the motion of the robot. This included a description of the hardware setup, the geometry involved, the methods used to estimate the displacement of image features, and the techniques used to help ensure the final motion estimate was reliable and robust against noise and erroneous feature matches.

Results from simulations then showed how the performance of the system varied with changes in system parameters, describing how these parameters both affect the range of motion able to be estimated and how they can be tuned to balance accuracy and reliability with speed.

Finally, physical experiments validated the system as a viable relative localisation solution. The error levels obtained are sufficiently low to warrant its use as a means of estimating the motion – and hence perform relative localisation – of a mobile robot moving over potentially rough and unpredictable and hence error-producing terrain.

Chapter 4

Obstacle Detection with Passive Stereo Vision

> “ *If you can find a path with no obstacles, if probably doesn't lead anywhere.* ”
>
> **Frank A. Clark**

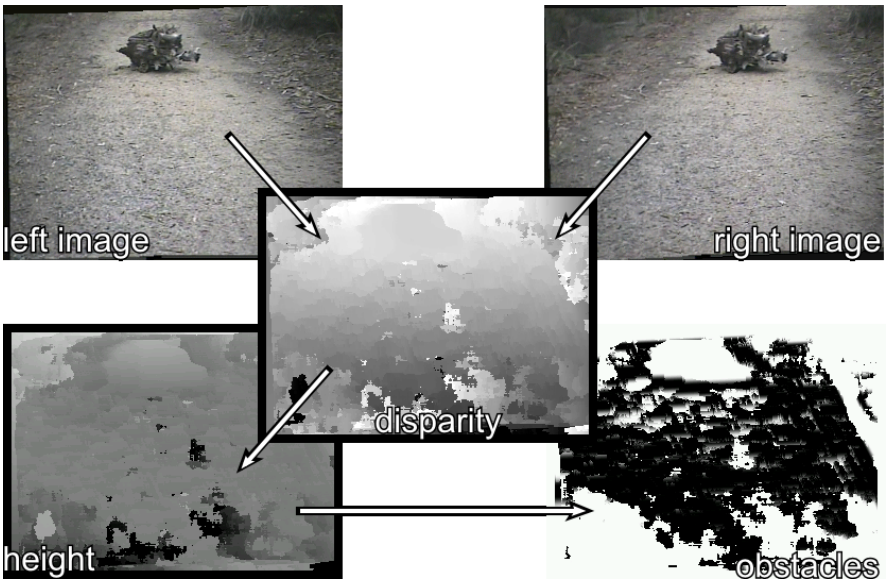


Figure 4.1: From images to obstacles

With the exception of trivial environments, all robots will at some point encounter an obstacle. In the most generalised sense, an obstacle is anything that serves to impede a robot's intended motion. Obstacles can be fixed objects, like walls or chairs indoors or trees or fences outside. Obstacles can also be moving objects, such as people, vehicles and indeed other robots. To ensure safe operation (for the robot, people and for potentially delicate

objects), and to complete the mission, obstacles must be avoided. In some scenarios, there may be no dynamic obstacles, and a robot may be able to rely upon high-quality *a priori* maps and high-quality localisation, meaning obstacles can simply be avoided with appropriate path-planning. More often, however, conditions are not so amenable, and so we must have a means of detecting obstacles first.

4.1 Introduction

In the bush, whether along tracks or during a cross-country traversal, obstacles generally take on one of two forms. The first set is perhaps the most obvious: trees, fallen branches, large rocks and embankments. All of these are objects or features of the terrain that extend *above* what is otherwise typically a fairly flat ground. The second set includes potholes, ditches and track-side drainage trenches. All of these obstacles extend *below* an otherwise flat ground. (Obstacles of this form are commonly referred to as "*negative obstacles*".) Both forms are equally able to trap or impede an unwary robot, so both forms must be able to be detected reliably.

Traditional forms of obstacle detection include sonar and LIDAR range-finders. Both of these range sensors have the capacity to provide highly accurate range and bearing readings at fairly high update rates. However, as discussed earlier in Section 1.4.2, both of these forms of sensing suffer from problems in bush environments that mean they are not the best choice, in spite of their widespread use elsewhere. Most notably – with the exception perhaps of some sophisticated and expensive sensor arrays – neither is able to provide a dense measurement, beyond a single plane at best, in a single reading. To counter this, the sensor must either be very carefully positioned to provide a hopefully usable reading most of the time (while still having a limited information density), or a mechanism must be incorporated (such as a tilt or pan-tilt unit) that allows the sensor to make a series of scans in different directions. Such a solution introduces many problems of physical stability, calibration and timing that can easily outweigh the perceived benefit of using sonar or LIDAR in the first place. Vision, and stereo vision in particular, is a viable alternative that in many ways can be superior due to its lower sensor cost and greater information density.

Stereo vision allows dense range maps to be created with good coverage in horizontal and vertical planes. The equipment requirements are also simple and cheap: a pair of cameras are all that is required, and, as mentioned earlier (Section 1.4.2) cameras are typically an order of magnitude lower in cost than LIDAR scanners. The biggest problem with stereo vision – a problem that is still under active investigation – is the problem of correspondence-matching (finding a feature defined in one image in a second image) and it's associated computational cost. In bush environments, the problem of correspondence is perhaps more easily overcome than in artificial environments: the visual richness and irregularity means simple image patches can be used effectively as features. In artificial environments, the large degree of visual uniformity usually means alternative features, such as corners or edges, must be used instead, resulting in a much lower-density depth-map.

A geometrically-focused mathematical review of stereo vision is provided in Appendix 8.2.6 for completeness, and the reader is advised to peruse this section first if they are unfamiliar with stereo systems.

The stereo vision system presented in this chapter makes use of well-established techniques to transform a pair of images into a depth-map that can then be used to create an obstacle

likelihood map, a map that depicts the likelihood that an area contains something that would impede traversal. The work presented does not aim to produce a better stereo algorithm but serves to provide a means through which a robot can find areas of safe traversal and hence become functional.

To this end, we choose to define obstacles as any part of the environment – whether distinct objects or parts of the terrain – that could present a hazard to the robot. The simple notion of deviation – that is, a difference in height – of terrain from an assumed flat ground is used to describe areas that are potentially unsafe. Through calibration we determine a reference ground plane that is fixed relative to the robot. In operation, we determine a local ground plane estimate that is constrained by the calibrated ground plane. This constraint ensures that we do not inadvertently take an unsafe local plane – such as the side of an embankment – as our new obstacle reference. Areas are then given a likelihood of danger based on their deviation from this local plane. This method implicitly allows both positive (above-ground) and negative (below-ground) obstacles to be detected. Furthermore, the potential hazard is made to be related to the (absolute) height difference between the terrain and the assumed ground: small deviations are tolerable while large deviations should be strongly avoided. The result is an obstacle likelihood map: a map that describes at each location the likelihood that that location contains a hazard.

4.2 The Stereo Rig

4.2.1 Physical Setup

The stereo rig consists of two identical cameras. Each is a standard $\frac{1}{3}-$inch CCD video camera with a stated sensor resolution of 537×597 pixels, producing an analog PAL signal. It is housed in a rugged security-style metal case and uses CS-mount lenses. The lens used is of 8.0 mm focal length, giving an effective field-of-view of approximately 32 degrees horizontally and 24 degrees vertically. The camera is coupled to a standard PC frame-grabber, specifically a FlyVIDEO 3000 PCI capture card, which uses the Philips SAA7134 chipset. Images of 320×240 pixels are used.

The cameras are rigidly attached to screw-adjustable brackets which in turn are rigidly attached to a mounting beam. For the purposes of setup and calibration, each camera has three degrees of limited rotational freedom and one of limited translation (along a vertical axis). The baseline – that is, the horizontal separation between cameras – can also be altered. Once adjusted, the cameras can be locked such that the pair remain rigidly coupled via the mounting beam. The beam itself is mounted to the robot rigidly but, due to the size and weight distribution of the beam-camera combination, vibration is possible. However, in spite of this allowed vibration, the cameras remain fixed without movement relative to each-other. Thus, even if the overall rig is bumped (within limits) or vibrates while the robot is traversing rough terrain, the extrinsic stereo parameters – the position of the right camera with respect to the left – remains unchanged, meaning time-consuming recalibration is not necessary.

Figure 4.2 depicts the forward-facing camera rig. The two outer-most cameras form the stereo pair. Figure 4.3 shows a close-up image of the adjustable individual camera mount, highlighting the three small screws to adjust the pitch, roll and height of the camera and the large screw to adjust the yaw of the camera.

The physical calibration of the rig aims to balance the desired working region (effectively the minimum and maximum usable range) with resolution (measurable disparity). With

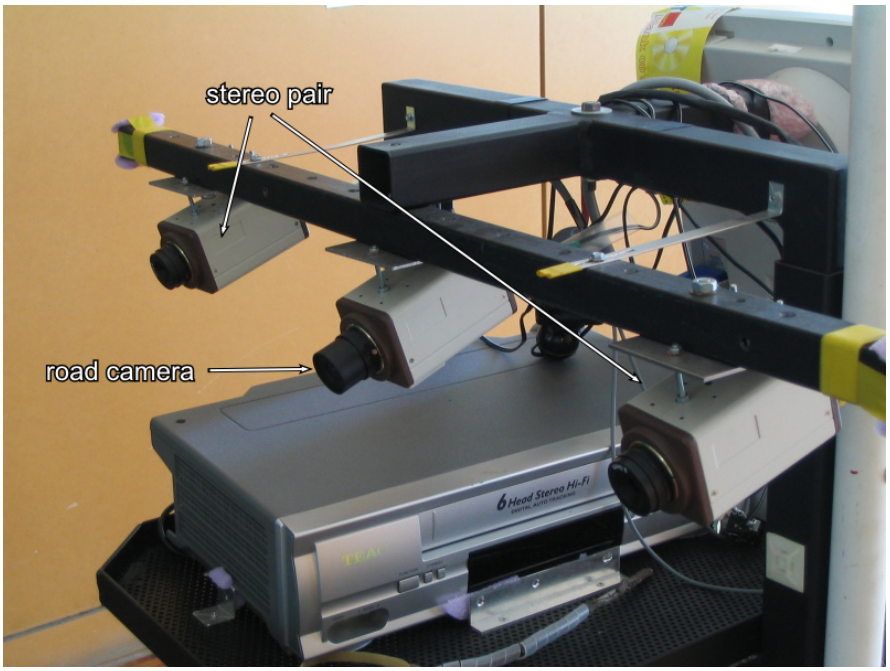


Figure 4.2: Multi-camera rig

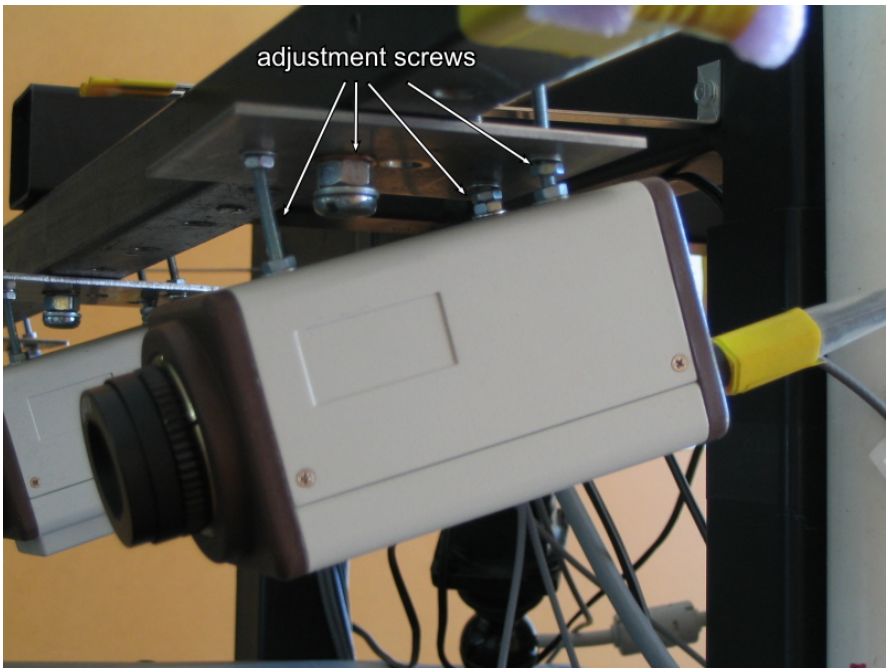


Figure 4.3: Camera mounting

the notion of path-planning in mind, the desired working region is the area in front of the robot centred a few metres ahead. To balance this desired working region against measurable disparity, a relatively wide baseline of 500 mm is used and the cameras converge at a point approximately 5000 mm ahead of the robot. This gives a ground-level blind-spot that extends to a distance of approximately 1735 mm in front of the robot.

This blind-spot could be reduced a number of ways, all of which are trade-offs. Firstly, the cameras could converge at a closer point, but this would reduce the maximum range and also increase viewpoint distortion (the difference in the appearance of objects from two different viewpoints), making the search for correspondences more error-prone. A narrower baseline could be used, but this would reduce the range of disparities and hence the range resolution. Wider-angle lenses could be used, but this would introduce greater lens distortion and also reduce depth-resolution (as each pixel would then correspond to a larger area). The cameras could be tilted down further, but this would reduce the maximum range.

The setup used is seen as an appropriate balance of parameters that provides a working region and resolution appropriate for the task.

The final element of physical calibration involves defining a common ground-plane, common, that is, to the stereo pair and the third forward-facing camera, (whose use for road-finding is described in Chapter 5). This step serves two purposes: firstly, it provides a predefined ground-plane which constrains stereo-derived local ground-plane estimates and secondly, it provides a common frame of reference that allows the results of stereo processing to be combined with the results of road-finding in order to perform local path-planning (as described in Chapter 6). This stage is shown in Figure 4.4, which also shows the blind area in front of the robot.

The front (far) calibration board shown in Figure 4.4 is used to define a virtual reference frame whose origin is shown in the figure. Calibration provides the position and orientation of this origin in each camera's reference frame. Knowing the location of a point in any image we can reproject that point, via this common reference frame, into a new, virtual reference frame that is common and consistent, allowing information derived from different cameras to be consistently combined.

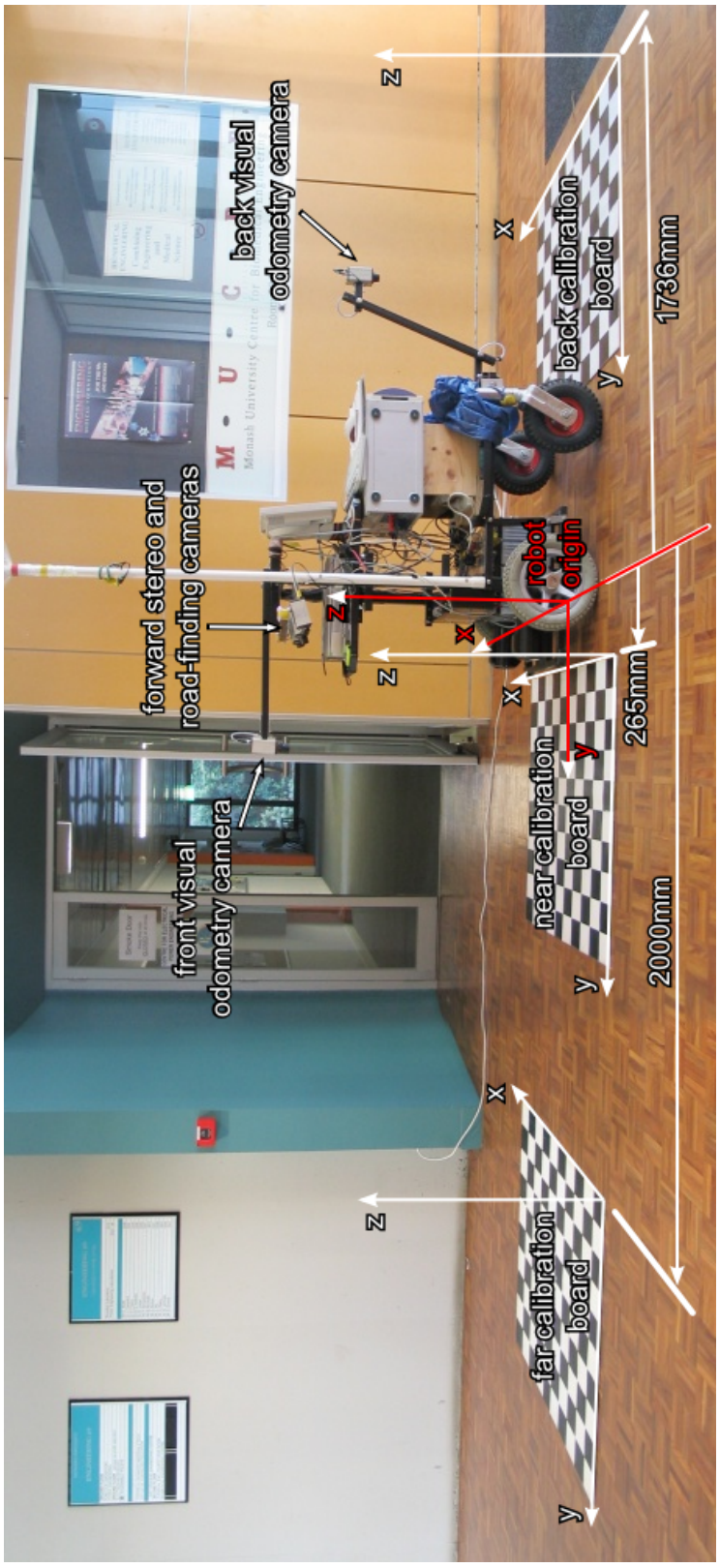


Figure 4.4: Joint camera calibration

4.2.2 Software Calibration

Software calibration is performed with the widely used "*Camera Calibration Toolbox for Matlab*" [10] (whose algorithms are also available through the "*Open Source Computer Vision Library*" [58]). The first stage of calibration is done to determine the intrinsic camera parameters such that images can be corrected for lens distortion. This is a crucial step for any image-processing operation where real-world, metric measurements need to be made. (Indeed, *all* cameras used in the work presented in this thesis are first similarly calibrated such that the images they produce can be corrected for lens distortion.)

The second stage of calibration determines the extrinsic parameters of the stereo rig in order to rectify the images such that epipolar lines are brought in alignment. This step, while not a requirement, is highly recommended and standard practice. By aligning epipolar lines, the correspondence search is reduced from a two-dimensional search through the image to a one-dimensional search along corresponding epipolar lines. This makes it possible to produce dense stereo maps relatively quickly. This stage produces both the set of extrinsic parameters (see Section .1 for details of these parameters) and two lookup tables, one for each camera. These lookup tables provide the pixel-based co-ordinate mappings necessary to rectify images and correct for both distortion (using the intrinsics estimated in the first stage) and epipolar alignment (using the extrinsics just estimated).

The final calibration stage involves determining the location and orientation of the assumed ground plane, relative to the stereo rig. The relationship between a point in the camera reference frame $\boldsymbol{p}_c$ and a point in another, known reference frame $\boldsymbol{p}_r$ is given by

$$\boldsymbol{p}_r = \mathsf{R}_c^{-1}(\boldsymbol{p}_c - \boldsymbol{T}_c) \tag{4.1}$$

where R_c and $\boldsymbol{T}_c$ are a rotation matrix and translation vector describing the orientation and position of the reference frame with respect to the camera. The final stage of calibration retrieves the parameters of a common reference frame, those being R_c and $\boldsymbol{T}_c$. The ground plane is then taken to be the $z = 0$ plane in the world reference frame.

This plane can be described by a (unit) normal vector to the plane $\hat{\mathbf{N}}_p$ and a point on the plane (which we will call its anchor) $\boldsymbol{A}_p$, both defined in the camera's reference frame. These can be determined from R_c and $\boldsymbol{T}_c$ via

$$\hat{\mathbf{N}}_p = \mathsf{R}_c \boldsymbol{z} \tag{4.2}$$

where $\boldsymbol{z}$ is the camera z-axis vector $[0, 0, 1]^T$ and

$$\hat{\mathbf{A}}_p = \boldsymbol{T}_c \tag{4.3}$$

These two vectors are used to determine the position of points with respect to the ground-plane.

4.3 Correspondence Matching

At this stage, we have available a pair of rectified images. The physical setup and following software correction mean that distortion has been removed from the images and the epipolar lines lie on corresponding rows of the images. Therefore, as desired, the search for a feature from left to right image is reduced to a one-dimensional search.

The feature chosen for correspondence matching is the simple notion of image patches in order to produce a dense stereo map. Alternatives features include edges and corners, although the use of such features, which are typically sparsely distributed through images, produce sparse depth maps. As discussed in Section 3.2, the visual richness and irregularity of bush environments makes raw patches of images a useful and reliable feature for matching.

We therefore need to look at how patches can be matched in order to solve the problem of stereo feature correspondence. A number of common similarity measures – the most common and easily implemented measure of whether two patches match – were presented in Section 3.5.2 and are repeated here for clarity:

A feature is described by a patch of an image, which we refer to as a template τ, defined by a centre point $[r_\tau, c_\tau]^T$, (row and column numbers) a width of $2C_\tau + 1$ columns and height $2R_\tau + 1$ rows. For a generalised similarity function $A \circledast B$ between a template in image A and an equally-sized patch in image B, centred at $[r', c']^T$, the correlation-based similarity is defined as

$$S(\tau, r', c') = \sum_{r=-R_\tau}^{R_\tau} \sum_{c=-C_\tau}^{C_\tau} A[r_\tau + r, c_\tau + c] \circledast B[r' + r, c' + c] \tag{4.4}$$

which can also be area-normalised as

$$S_\eta(\tau, r', c') = \frac{S(\tau, r', c')}{\eta} \tag{4.5}$$

with normalising factor

$$\eta = \sum_{r=-R_\tau}^{R_\tau} \sum_{c=-C_\tau}^{C_\tau} |A[r_\tau + r, c_\tau + c]| \times \sum_{r=-R_\tau}^{R_\tau} \sum_{c=-C_\tau}^{C_\tau} |B[r' + r, c' + c]| \tag{4.6}$$

Three of the most common similarity functions are the absolute difference

$$A \circledast B = \Big|A - B\Big| \tag{4.7}$$

the squared difference

$$A \circledast B = \Big(A - B\Big)^2 \tag{4.8}$$

and the product

$$A \circledast B = A \times B \tag{4.9}$$

with the addition of mean-normalised versions substituting $\widehat{A}$ for A and $\widehat{B}$ for B, where

$$\widehat{A} = A - \bar{A} \tag{4.10}$$

$$\bar{A} = \frac{1}{(2R_\tau + 1)(2C_\tau + 1)} \sum_{r=-R_\tau}^{R_\tau} \sum_{c=-C_\tau}^{C_\tau} |A[r_\tau + r, c_\tau + c]|$$

with $\widehat{B}$ defined similarly.

The factors to be considered when selecting a measure are different with a stereo setup, compared to the single-camera visual odometry system described earlier. With a stereo setup, we have two independent image acquisition systems (in this case each consisting of a camera,

cabling and frame-grabber) and moderately different viewpoints. However, in the context of usage, the distance of objects within the scene will typically be large compared to the camera separation, meaning differences in image intensity due to differing viewpoints will be small. Any differences in image intensity will be primarily due to differences in hardware – essentially differences in camera gains – which we can assume will remain constant over time and across the image. If required, differences in camera gains can be compensated for through a normalising pre-processing stage (however, with both cameras set to fixed-gain, the difference was found to be small enough to be ignored). Therefore, mean-normalised similarity measures (utilising Eq^n 4.10) are unnecessary, leading to computational savings.

In comparing the three presented similarity measures, we first examine Eq^n 4.9, commonly known as the cross-correlation. When fully normalised (as is the case when using Eq^n 4.5 and Eq^n 4.10), the normalised cross-correlation is a good measure of similarity and is highly robust against differences in image intensities and contrast levels. When *not* normalised, it does not perform well and can be somewhat sensitive to the data itself, and not just the *similarity* of the data. As an extreme example, if A consisted entirely of zeros, the un-normalised cross-correlation would be zero regardless of the contents of B, which is clearly a poor indication of the similarity of the two. With the left and right images expected and assumed to have comparable illumination, the intensity invariance of normalised cross-correlation is of limited benefit. The need for costly normalisation to obtain reasonable results is enough to rule out this measure. We are then left with Eq^n 4.7 and Eq^n 4.8, neither of which require area-normalisation to perform to an adequate level. These two measures perform very similarly and there is no great reason to choose one over the other. In order to fall on the side of computational efficiency, we will select Eq^n 4.7 which offers a slight reduction in computation (an absolute operation as compared to a multiplication).

Without the need for mean-normalisation and with rectified images available, the search for feature correspondences can be simplified, resulting in an efficient algorithm. The search for correspondences is identical to the template-matching problem presented earlier in Section 3.5.2, with the addition of a further constraint. To reiterate, (with modifications specific to the correspondence problem and to the similarity measure chosen) for an absolute difference similarity function $\left|I_L - I_R\right|$ between a template τ in image I_L centred at $[r_\tau, c_\tau]^T$ and an equally-sized patch in image I_R centred at $[r', c']^T$, correlation-based similarity is defined as

$$S(r_\tau, c_\tau, r', c') = \sum_{r=-R_\tau}^{R_\tau} \sum_{c=-C_\tau}^{C_\tau} \left| I_L[r_\tau + r, c_\tau + c] - I_R[r' + r, c' + c] \right| \tag{4.11}$$

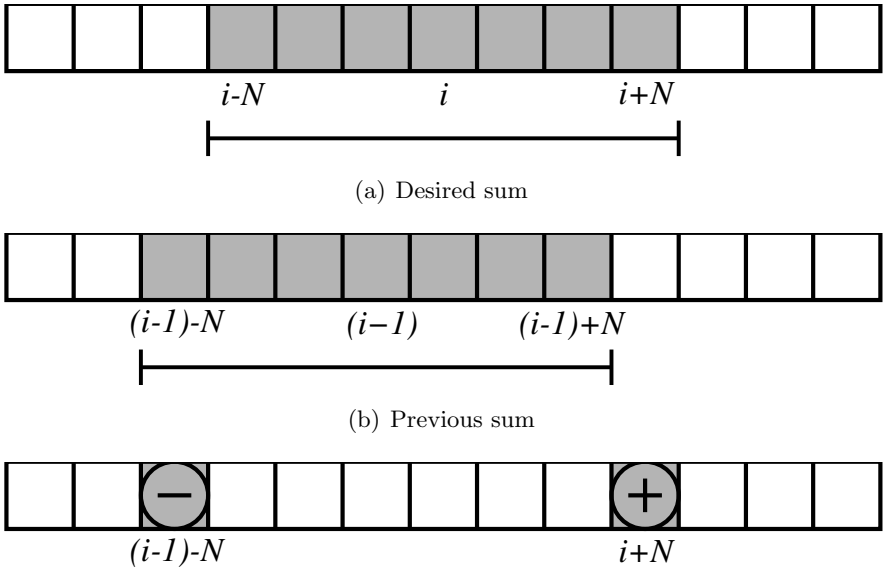
The correspondence problem consists of finding the $[r', c']^T$ that minimises the function S, subject to the epipolar and window constraints:

$$\begin{aligned} r' &= r_\tau \\ c' &\in \{c_\tau - C_\omega, \ldots, c_\tau + C_\omega\} \end{aligned}$$

for a search window of width $2C_\omega + 1$.

That is, the search is restricted to be along the same image rows in left and right images, and the search window is centred on the left-image column of the template.

The disparity is then simply defined as $d = c_\tau - c'$ and we can restate Eq^n 4.11 in terms



(a) Desired sum

(b) Previous sum

(c) Corrections

Figure 4.5: Optimised sequential summation

of disparity as:

$$S(d, r_\tau, c_\tau) = \sum_{r=-R_\tau}^{R_\tau} \sum_{c=-C_\tau}^{C_\tau} \Big| I_L[r_\tau + r, c_\tau + c] - I_R[r_\tau + r, c_\tau + c - d] \Big| \tag{4.12}$$

with disparity typically being restricted to

$$d \in \{0, \ldots, D_{max}\}$$

As can be seen, the epipolar constraint combined with rectified images reduces the search to one dimension, along image rows.

4.3.1 Optimisation through Dynamic Programming

When constructing a dense disparity map, there is a great opportunity for optimisation through the application of dynamic programming principles, as shown, for example in [33]. At a given level of disparity, there is much overlap in evaluation of S for adjacent templates. Figure 4.5 graphically depicts a simplified situation of a one-dimensional summation and a possible optimisation. Figure 4.5(a) shows the desired summation of a sequence of elements, centred on i and extending from $i - N$ to $i + N$. The elements involved in the summation are shaded. This summation can be equivalently composed of a (previously calculated) summation centred on $(i - 1)$ (shown in Figure 4.5(b)), and two corrections: the subtraction of the element at $(i - 1) - N$ and the addition of the element at $i + N$ (shown in Figure 4.5(c)). In this manner, we can reuse previously calculated results and significantly reduce redundant operations.

This optimisation can also be shown mathematically as follows: In general, for a function $g(i)$ with $i \in \mathbb{Z}$ defined as

$$g(i) = \sum_{j=-N}^{+N} f(i+j)$$

we can make the following recursive expansion:

$$\begin{aligned} g(i) &= \sum_{j=-N}^{+N} f(i+j) \\ &= \left(\sum_{j=-N}^{+N} f\Big((i-1)+j\Big) \right) - f\Big((i-1)-N\Big) + f(i+N) \\ &= g(i-1) - f(i-1-N) + f(i+N) \end{aligned} \tag{4.13}$$

where the first two terms make use of previous calculations.

This can be applied to Eq^n 4.12. Firstly we define the following:

$$f_S(d, r_f, c_f) = I_L[r_f, c_f] \circledast I_R[r_f, c_f - d] \tag{4.14}$$

$$g_S(d, r_g, c_g) = \sum_{c=-C_\tau}^{C_\tau} f_S(d, r_g, c_g + c) \tag{4.15}$$

$$h_S(d, r_h, c_h) = \sum_{r=-R_\tau}^{R_\tau} g_S(d, r_h + r, c_h) \tag{4.16}$$

Therefore, from Eq^n 4.12, we simply have

$$S(d, r_\tau, c_\tau) = h_S(d, r_\tau, c_\tau) \tag{4.17}$$

Applying Eq^n 4.13 we then have

$$\begin{aligned} g_S(d, r_g, c_g) &= \sum_{c=-C_\tau}^{C_\tau} f_S(d, r_g, c_g + c) \\ &= g_S(d, r_g, c_g - 1) - f_S(d, r_g, c_g - 1 - C_\tau) + f_S(d, r_g, c_g + C_\tau) \end{aligned} \tag{4.18}$$

and

$$\begin{aligned} h_S(d, r_h, c_h) &= \sum_{r=-R_\tau}^{R_\tau} g_S(d, r_h + r, c_h) \\ &= h(d, r_h - 1, c_h) - g_S(d, r_h - 1 - R_\tau, c_h) + g_S(d, r_h + R_\tau, c_h) \\ &= h(d, r_h - 1, c_h) - g_S(d, r_h - 1 - R_\tau, c_h) \\ &\quad + g_S(d, r_h + R_\tau, c_h - 1) - f_S(d, r_h + R_\tau, c_h - 1 - C_\tau) \\ &\quad + f_S(d, r_h + R_\tau, c_h + C_\tau) \end{aligned} \tag{4.19}$$

In Eq^n 4.19 all but the last term will be a value that has been previously evaluated for previous calculations (with the exceptions of the special cases of the first row or column). As such, we can make heavy use of previous results to improve efficiency and performance.

The result is that we can significantly reduce the number of summations required by reusing previously calculated results, hence speeding up the otherwise very costly calculation of similarity. Application of Eq^n 4.13 allows a rapid evaluation of similarity at each point for a *given disparity*. We therefore repeat this process for each allowable disparity, keeping track of the best similarity (and corresponding disparity) found thus far.

4.3.2 Correspondence Matching Algorithms

The final correspondence-matching procedure then consists of two parts: calculation of similarities (Algorithm 4.1) and determination of the most likely disparity subject to a minimum similarity threshold (Algorithm 4.2).

4.4 Local Adaptability

In many environments, including bush environments, there exist many variations in the local terrain that, while varying from the calibrated and ideal ground plane, would not serve to restrict or hinder a robot's motion. These variations are localised, average deviations in the terrain that result from, for example, the robot approaching a traversable incline or decline, such as would be seen when at the top or bottom of a ramp. In these cases, there is a definite deviation of the terrain from the plane on which the robot sits (which is the assumed ground plane), but there is not necessarily any risk associated with traversal.

In order to adapt to these local changes, we introduce a local ground-plane estimation stage into the obstacle detection system. Application of the algorithms above yields a disparity image which can easily be transformed into a depth-map utilising the ideas presented in Section .5. At this point we have available a two-and-a-half dimensional (viewpoint-restricted) depth-map that describes the three-dimensional position of each point in the image within the camera's reference frame. We also have, after calibration, a concise, vector-based description of an ideal planar ground surface. It is these two sets of data – locally sensed terrain data and a calibrated ground description – that are used to determine a local ground-plane estimation.

The method uses the RANSAC algorithm [37] to fit a plane to the stereo-derived set of three-dimensional terrain points. The underlying assumption is that the ground is essentially locally planar and that most of what is seen in the image is the ground (such that points that lie on the ground are in the majority and are considered inliers by the RANSAC algorithm).

A subset of the terrain points is used in order to both reduce the amount of data to be processed and to provide a better sample of the ground (that is, to exclude what may potentially be nearby foliage other obstacles). Specifically, a window of points located just in front of the robot is used to determine the ground plane. In Figure 4.6 the sampled region is shown bounded by a rectangle.

The model we are trying to fit to our data is a plane, described by a (unit) normal vector $\hat{\mathbf{N}}_p$ and an anchor point $\boldsymbol{A}_p$. Any three (non-collinear) points $\boldsymbol{p}_a$, $\boldsymbol{p}_b$ and $\boldsymbol{p}_c$ can be used to define the plane as follows:

$$\hat{\mathbf{N}}_p = \frac{(\boldsymbol{p}_b - \boldsymbol{p}_a) \times (\boldsymbol{p}_c - \boldsymbol{p}_a)}{\|(\boldsymbol{p}_b - \boldsymbol{p}_a) \times (\boldsymbol{p}_c - \boldsymbol{p}_a)\|} \tag{4.20}$$

Algorithm 4.1: Stereo correspondence similarity calculation

Procedure: S = `Similarity(`I_L`,` I_R`,` d`,` C_τ`,` R_τ`)`

Input: left and right images I_L, I_R
Input: disparity d
Input: patch dimensions C_τ and R_τ, where $width = 2C_\tau + 1$ and $height = 2R_\tau + 1$

Output: similarity array S

Data: ROWS, COLS ▷ `image dimensions`
Data: $R_{min} = R_\tau$, $R_{max} = \text{ROWS} - 1 - R_\tau$ ▷ `row limits`
Data: $C_{min} = C_\tau$, $C_{max} = \text{COLS} - 1 - C_\tau$ ▷ `column limits`
Data: F_S, G_S, H_S ▷ `intermediate data`

▷ `first column special case for` G_S
for $r' \in \{R_{min} - R_\tau, \ldots, R_{max} + R_\tau\}$ **do**
  for $c \in \{-C_\tau, \ldots, C_\tau\}$ **do**
    $F_S[r', c] \leftarrow I_L[r', c] \circledast I_R[r', c - d]$
    $G_S[r', C_{min}] \overset{+}{\leftarrow} F_S[r', c]$
  end
end

▷ `first row, first column` ***extra*** `special case for` H_S
$H_S[R_{min}, C_{min}] \leftarrow \sum_{r=-R_\tau}^{R_\tau} G_S[R_{min} + r, C_{min}]$
$S[R_{min}, C_{min}] \leftarrow H_S[R_{min}, C_{min}]$

▷ `first column remaining rows special case for` H_S
for $r' \in \{R_{min} + 1, \ldots, R_{max}\}$ **do**
  $H_S[r', C_{min}] \leftarrow H_S[r' - 1, C_{min}] - G_S[r' - 1 - R_\tau, C_{min}] + G_S[r' + R_\tau, C_{min}]$
  $S[r', C_{min}] \leftarrow H_S[r', C_{min}]$
end

▷ `first row remaining columns special case for` H_S
for $c' \in \{C_{min} + 1, \ldots, C_{max}\}$ **do**
  $H_S[R_{min}, c'] \leftarrow 0$
  for $r \in \{-R_\tau, \ldots, -R_\tau\}$ **do**
    $F_S[R_{min} + r, c' + C_\tau] \leftarrow I_L[R_{min} + r, c' + C_\tau] \circledast I_R[R_{min} + r, c' + C_\tau - d]$
    $G_S[R_{min} + r, c'] \leftarrow$
    $G_S[R_{min} + r, c' - 1] - F_S[R_{min} + r, c' - 1 - C_\tau] + F_S[R_{min} + r, c' + C_\tau]$
    $H_S[R_{min}, c'] \overset{+}{\leftarrow} G_S[R_{min} + r, c']$
  end
  $S[R_{min}, c'] \leftarrow H_S[R_{min}, c']$
end

Algorithm 4.1: continued

▷ `remaining rows`
for $r' \in \{R_{min}+1, \ldots, R_{max}\}$ **do**
  for $c' \in \{C_{min}, \ldots, C_{max}\}$ **do**
    $F_S[r'+R_\tau, c'+C_\tau] \leftarrow I_L[r'+R_\tau, c'+C_\tau] \circledast I_R[r'+R_\tau, c'+C_\tau-d]$
    $G_S[r'+R_\tau, c'] \leftarrow G_S[r'+R_\tau, c'-1] - F_S[r'+R_\tau, c'-1-C_\tau] + F_S[r'+R_\tau, c'+C_\tau]$
    $H_S[r', c'] \leftarrow H_S[r'-1, c'] - G_S[r'-1-R_\tau, c'] + G_S[r'+R_\tau, c']$
    $S[r', c'] \leftarrow H_S[r', c']$
  end
end

Algorithm 4.2: Image similarity estimation

Procedure: `D = Disparity(`I_L`,` I_R`,` C_τ`,` R_τ`,` S_{min}`)`

Input: left image I_L, right image I_R
Input: patch dimensions C_τ and R_τ, where $width = 2C_\tau + 1$ and $height = 2R_\tau + 1$
Input: minimum allowed similarity S_{min}

Output: disparity image D

Data: ROWS, COLS ▷ `image dimensions`
Data: $R_{min} = R_\tau$, $R_{max} = \text{ROWS} - 1 - R_\tau$ ▷ `row limits`
Data: $C_{min} = C_\tau$, $C_{max} = \text{COLS} - 1 - C_\tau$ ▷ `column limits`
Data: S ▷ `set of similarity arrays`

▷ `construct similarity arrays`
for $d \in \{0, \ldots, D_{max}\}$ **do**
  $S[d] \leftarrow$ `Similarity`$(I_L, I_R, d, C_\tau, R_\tau)$
end
▷ `find most likely disparities (those with the greatest similarity)`
for $r \in \{R_{min}, \ldots, R_{max}\}$ **do**
  for $c \in \{C_{min}, \ldots, C_{max}\}$ **do**
    $S_{best} \leftarrow S_{min}$
    $d_{best} \leftarrow \infty$
    for $d \in \{0, \ldots, D_{max}\}$ **do**
      if $S[d][r, c] > S_{best}$ **then**
        $S_{best} \leftarrow S[d][r, c]$
        $d_{best} \leftarrow d$
      end
    end
    $D[r, c] \leftarrow d_{best}$
  end
end

Figure 4.6: Sampled ground region

and

$$\boldsymbol{A}_p = \frac{\boldsymbol{p}_a + \boldsymbol{p}_b + \boldsymbol{p}_c}{3} \tag{4.21}$$

To ensure only sensible planes are used a normal constraint is included. For example, if the robot were facing a wall the recovered plane could easily be the vertical plane of the wall, which is clearly an obstacle and should *not* be used as a reference. The normal constraint is simply an upper threshold on the allowed angle between a recovered normal and the calibrated normal, where the angle between two (unit) normal vectors is defined by

$$\theta = \text{Cos}^{-1}(\hat{\mathbf{N}}_{cal} \cdot \hat{\mathbf{N}}_{est}) \tag{4.22}$$

Essentially, the calibrated normal and the threshold angle, θ_{norm} describe a cone (shown in Figure 4.7) and for a plane to be considered viable its normal must lie within this cone.

To determine the fitness of a point $\boldsymbol{p}$ to the plane, we use the minimum distance, h between the point and the plane, given by

$$h = (\boldsymbol{p} - \boldsymbol{A}_p) \cdot \hat{\mathbf{N}}_p \tag{4.23}$$

We make use of Eq^n 4.20, Eq^n 4.21 and Eq^n 4.23 in the RANSAC algorithm to define and test potential planes, with the result being a set of M points (inliers) that can then be used to fit a final plane model. For a set of M points $\{\boldsymbol{p}_0, \ldots, \boldsymbol{p}_{M-1}\}$ (with $M >= 3$) we make use of an alternative definition of a plane. Firstly, we define the anchor in terms of the normal as

$$\boldsymbol{A}_p = a\hat{\mathbf{N}}_p \tag{4.24}$$

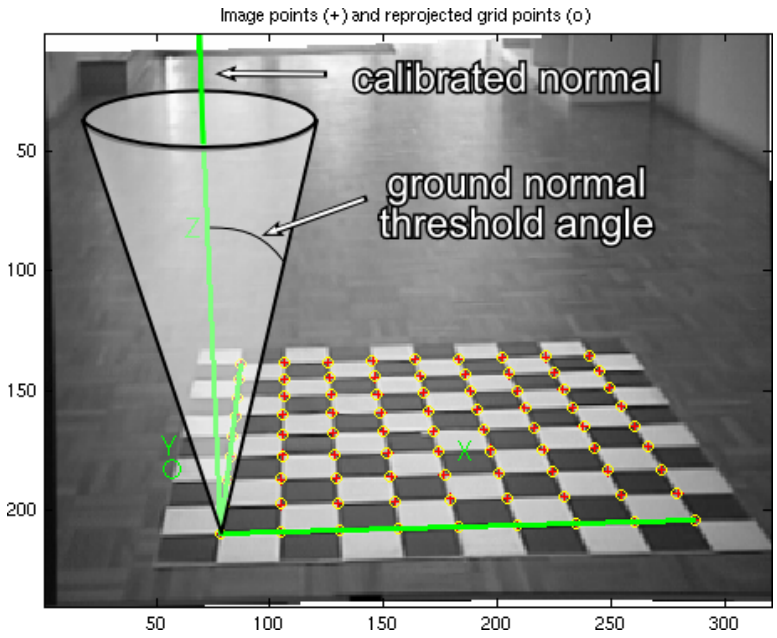


Figure 4.7: Ground normal constraint cone

Any point on the plane satisfies the equation

$$\begin{aligned} & (\boldsymbol{p} - \boldsymbol{A}_p) \cdot \hat{\mathbf{N}}_p = 0 \\ \implies & (\boldsymbol{p} - a\hat{\mathbf{N}}_p) \cdot \hat{\mathbf{N}}_p = 0 \\ \implies & \boldsymbol{p} \cdot \hat{\mathbf{N}}_p - a\hat{\mathbf{N}}_p \cdot \hat{\mathbf{N}}_p = 0 \\ \implies & \boldsymbol{p} \cdot \hat{\mathbf{N}}_p - a = 0 \\ \implies & \boldsymbol{p} \cdot \hat{\mathbf{N}}_p = a \\ \implies & \boldsymbol{p} \cdot \boldsymbol{X} = 1 \end{aligned} \tag{4.25}$$

where

$$\boldsymbol{X} = \frac{1}{a}\hat{\mathbf{N}}_p \tag{4.26}$$

This gives

$$\begin{bmatrix} x_0 & y_0 & z_0 \\ \vdots & \vdots & \vdots \\ x_{M-1} & y_{M-1} & z_{M-1} \end{bmatrix} \begin{bmatrix} X_x \\ X_y \\ X_z \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \tag{4.27}$$

We can then solve Eq^n 4.27 for $\boldsymbol{X}$ in the least-squares sense using SVD. We can then recover $\hat{\mathbf{N}}_p$ and $\boldsymbol{A}_p$ using Eq^n 4.26 noting that $\hat{\mathbf{N}}_p$ is subject to the constraint of having unit magnitude.

The algorithm used to estimate the ground plane parameters is shown in Algorithm 4.3.

Algorithm 4.3: RANSAC-based ground-estimation algorithm

Procedure: $\{\hat{\mathbf{N}}_p, \boldsymbol{A}_p\}$ = `EstimateGround(`P_{raw}`,` $\hat{\mathbf{N}}_{cal}$`,` θ_{norm}`)`

Input: set of N stereo-derived points $P_{raw} = \{\boldsymbol{p}_0, \ldots, \boldsymbol{p}_{N-1}\}$
Input: calibrated ground plane normal $\hat{\mathbf{N}}_{cal}$ and threshold angle θ_{norm}
Output: estimated ground plane parameters $\hat{\mathbf{N}}_p$ and $\boldsymbol{A}_p$

Data: P_{best} ▷ set of M good points where $M \leq N$
Data: $\boldsymbol{p}$ ▷ point
Data: ε, E ▷ error, total error
Data: K ▷ number of inliers

1: $\bar{\varepsilon}_{best} \leftarrow \infty$ ▷ initialise best mean error
2: **for** *number of trials* **do**
3: **repeat**
4: $\{\boldsymbol{p}_a, \boldsymbol{p}_b, \boldsymbol{p}_c\} \leftarrow$ `RandomSelect`$(P_{raw}, 3)$ ▷ randomly select three points
5: **until** $\|(\boldsymbol{p}_b - \boldsymbol{p}_a) \times (\boldsymbol{p}_c - \boldsymbol{p}_a)\| \neq 0$ ▷ ensure points are not collinear
6: $\hat{\mathbf{N}}_{trial} \leftarrow \dfrac{(\boldsymbol{p}_b - \boldsymbol{p}_a) \times (\boldsymbol{p}_c - \boldsymbol{p}_a)}{\|(\boldsymbol{p}_b - \boldsymbol{p}_a) \times (\boldsymbol{p}_c - \boldsymbol{p}_a)\|}$ ▷ calculate normal (Eq^n 4.20)
▷ only consider estimated normals that lie within the defined cone
7: $\theta_{trial} \leftarrow \mathrm{Cos}^{-1}(\hat{\mathbf{N}}_{cal} \cdot \hat{\mathbf{N}}_{trial})$
8: **if** $\theta_{trial} > \theta_{norm}$ **then**
9: **goto** *3*
10: **end**
11: $\boldsymbol{A}_{trial} \leftarrow \dfrac{\boldsymbol{p}_a + \boldsymbol{p}_b + \boldsymbol{p}_c}{3}$ ▷ calculate anchor (Eq^n 4.21)
12: $K_{trial} \leftarrow 0$ ▷ initialise number of inliers
13: $P_{trial} \leftarrow \{\emptyset\}$ ▷ initialise point set
14: $E_{trial} \leftarrow 0$ ▷ initialise total error
▷ test each point against plane estimate
15: **for** $i \in \{0, \ldots, N-1\}$ **do**
16: $h \leftarrow (\boldsymbol{p}_i - \boldsymbol{A}_{trial}) \cdot \hat{\mathbf{N}}_{trial}$ ▷ calculate distance from plane (Eq^n 4.23)
17: $\varepsilon_i \leftarrow |h|$ ▷ determine error
18: **if** $\varepsilon_i < \varepsilon_{max}$ **then** ▷ compare error to threshold
19: $K_{trial} \overset{+}{\leftarrow} 1$ ▷ increment number of inliers
20: $P_{trial} \overset{\cup}{\leftarrow} \boldsymbol{p}_i$ ▷ add point to set of good points
21: $E_{trial} \overset{+}{\leftarrow} \varepsilon_i$ ▷ update total error
22: **end**
23: **end**
24: $\bar{\varepsilon}_{trial} \leftarrow E_{trial}/K_{trial}$ ▷ calculate mean error
25: **if** $\bar{\varepsilon}_{trial} < \bar{\varepsilon}_{best}$ **then** ▷ compare error against best error
26: $\bar{\varepsilon}_{best} \leftarrow \bar{\varepsilon}_{trial}$ ▷ store new best error
27: $P_{best} \leftarrow P_{trial}$ ▷ store new best point set
28: $K_{best} \leftarrow K_{trial}$ ▷ store new best number of points
29: **end**
30: **end**

Algorithm 4.3: continued

▷ P_{best} now contains the most consistent set of points

31: $\boldsymbol{X} \leftarrow \texttt{SVDSolve}(P_{best})$ ▷ solve Eq^n 4.27 for $\boldsymbol{X}$

▷ recover plane parameters

32: $a \leftarrow \frac{1}{\|\boldsymbol{X}\|}$

33: $\hat{\mathbf{N}}_p \leftarrow a\boldsymbol{X}$

34: $\boldsymbol{A}_p \leftarrow a\hat{\mathbf{N}}_p$

4.5 Locating Obstacles

We now have a depth-map that describes the three-dimensional position of each point in the reference image (the left stereo image) and an estimate of the local ground-plane. In the context of outdoor navigation in bush environments, we define obstacles to be regions that would impede the robot's motion, not, as is often done in other environments, as specific and discernable objects. Following this, we define obstacle regions as those that deviate significantly from an otherwise planar ground. This implicitly covers both *positive* obstacles (those extending above the ground, like trees, rocks, ridges or even overhanging vegetation) and *negative* obstacles (those extending below the ground, like potholes and ruts).

The process of evaluating terrain (that is, assigning points in the height-map a likelihood of there being an obstacle) consists of a mapping of the absolute height of terrain points compared to the ground, defined by a (unit) normal vector to the plane $\hat{\mathbf{N}}_p$ and an anchor point $\boldsymbol{A}_p$. Using the minimum distance, h, between the point and the plane (given by Eq^n 4.23) we then define the ground divergence d_g of a point as

$$d_g = |h| \tag{4.28}$$

The ground-based obstacle likelihood O_g, ranging from 0 (clear ground) to 1 (highest likelihood of an obstacle) is

$$O_g(d_g) = \begin{cases} 0 & , d_g < d_{floor} \\ \frac{1}{2} \cdot \left(1 - \cos\left(\frac{\pi(d_g - d_{floor})}{d_{ceil} - d_{floor}}\right)\right) & , d_{floor} \leq d_g < d_{ceil} \\ 1 & , d_g \geq d_{ceil} \end{cases} \tag{4.29}$$

where d_{floor} is a minimum divergence, below which points are assumed to be safe and hence given a zero obstacle score, and d_{ceil} is a maximum divergence, above which divergences carry maximum penalty (greatest likelihood of an obstacle). Figure 4.8 depicts this relationship graphically.

The specific form of this mapping function is not critical, however. The key elements are an increase in likelihood of an obstacle with increasing divergence from planar and a maximum allowable divergence. The idea is that, above this threshold, it no longer matters how much of divergence there is, it is simply enough to assign the highest likelihood of there being an impediment to motion. The divergence thresholds used are $d_{floor} = 25$, $d_{ceil} = 100$, empirically determined.

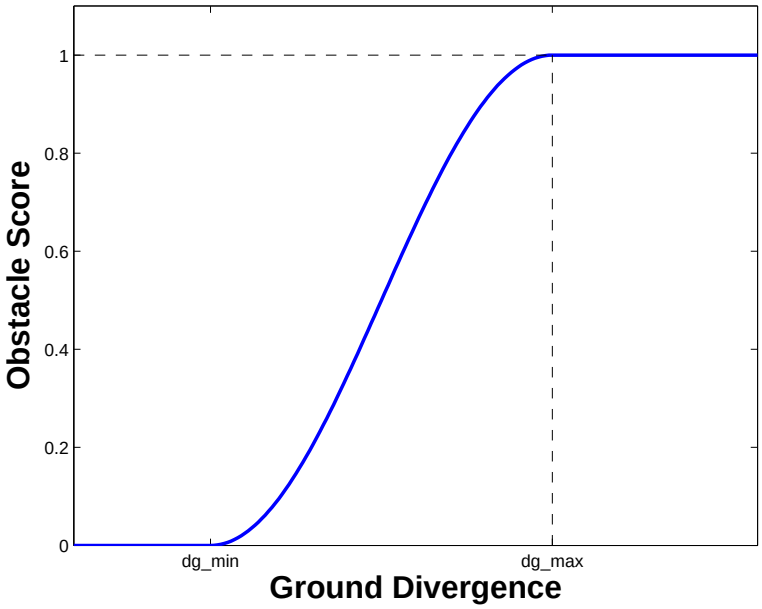


Figure 4.8: Obstacle mapping function

As a final point it should be noted that the default map state is one of high obstacle likelihood, that is, a pessimistic approach that errs on the side of caution. Accordingly, if the stereo-processing yields no information (typically because no correspondence match of high enough quality is found), we assume maximum likelihood.

4.6 Experimental Results

In this section, we present some experimental results that show the operation of the stereo system. In the following figures, the top left image shows the rectified left input image with the ground-sampling region shown as a rectangle. The top right image shows the resulting disparity image, the bottom left the height map (showing the height of each point relative to the estimated groundplane) and the bottom right image shows the final obstacle map. The disparity maps show low disparity as dark and high disparity as light, with black indicating an unknown region (essentially a region that was not able to be matched). The height maps' grey-scale indicates the height from black (low or negative z-values with respect to the ground plane) to white. Additionally, regions outside the height limits of what is displayed are clamped to either black or white. In the obstacle maps, dark regions indicate safe areas, that is, areas whose divergence from the estimated ground plane is small, while light regions depict areas that are either known to be unsafe (because they deviate too much from the groundplane) or are simply unknown (due to distance or gaps in the disparity map, for example) and hence pessimistically labelled as unsafe. Note also that the scaling of grey levels shown in the height maps is not consistent across the results shown. Where appropriate, levels have been scaled to highlight the underlying data. However, unless otherwise stated, the disparity map and obstacle map *are* consistently scaled. The parameters used are $d_{floor} = 25$,

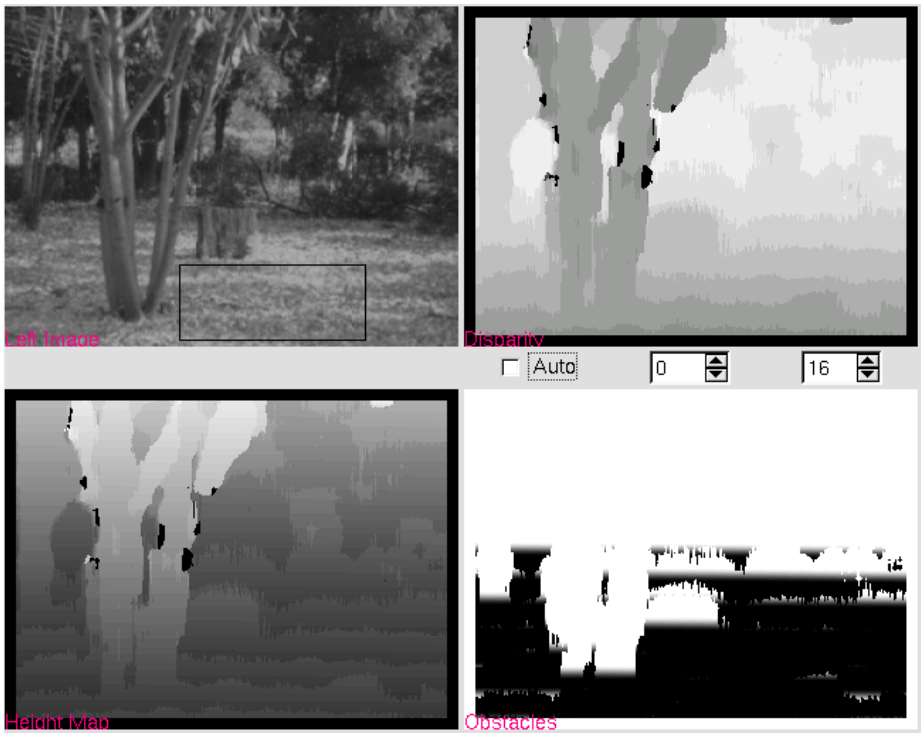


Figure 4.9: Tree test scene

$d_{ceil} = 100$, $\theta_{norm} = 15°$, $disparity \in [-64, +64]$, $S_{min} = 32$, $C_\tau = 8$, $R_\tau = 8$.

The first result, Figure 4.9 uses a standard pair of rectified stereo images, taken from the image database of Carnegie Mellon University's Vision & Autonomous Systems Center [19]. As the stereo system extrinsics are not the same as those of the robot, disparity and obstacle parameters have been adjusted. As shown in the obstacle map, the trees in the scene, as well as the tree-stump in the middle-ground, are all correctly labelled as obstacles, while the flat surrounding ground is generally considered safe.

In Figure 4.10 we see a fairly uneventful scene. The ground ahead is quite flat with little variation and is clearly traversable. Accordingly, the obstacle map shows safe ground essentially everywhere. Exceptions are the few small patches in the lower left and right corners (which, as shown in the disparity image are due to poor matches, most likely caused by the overly-bright patch on the left and the highly shadowed patch on the right). We also see a region at the top that is given a high obstacle likelihood. The height-map shows this region to be at a lower height (darker) than the rest of the ground, hence the classification of being likely to be an obstacle. As the robot progressed, however, the ground-plane estimation would adapt to the small change in terrain and so should then consider that region safe.

Figure 4.11 shows the approach to an aqueduct. The ground directly in front is fairly flat but drops away sharply to the upper right (two tree-trunks on the opposite face are visible in the top-right corner of the raw image). Accordingly, the obstacle map shows the ground in front to be safe but clearly indicates the danger posed by the aqueduct.

The scene in Figure 4.12 shows a clear and safe path ahead that forks to the left. The left fork also slopes down from the main track, and this is shown in the obstacle map: the

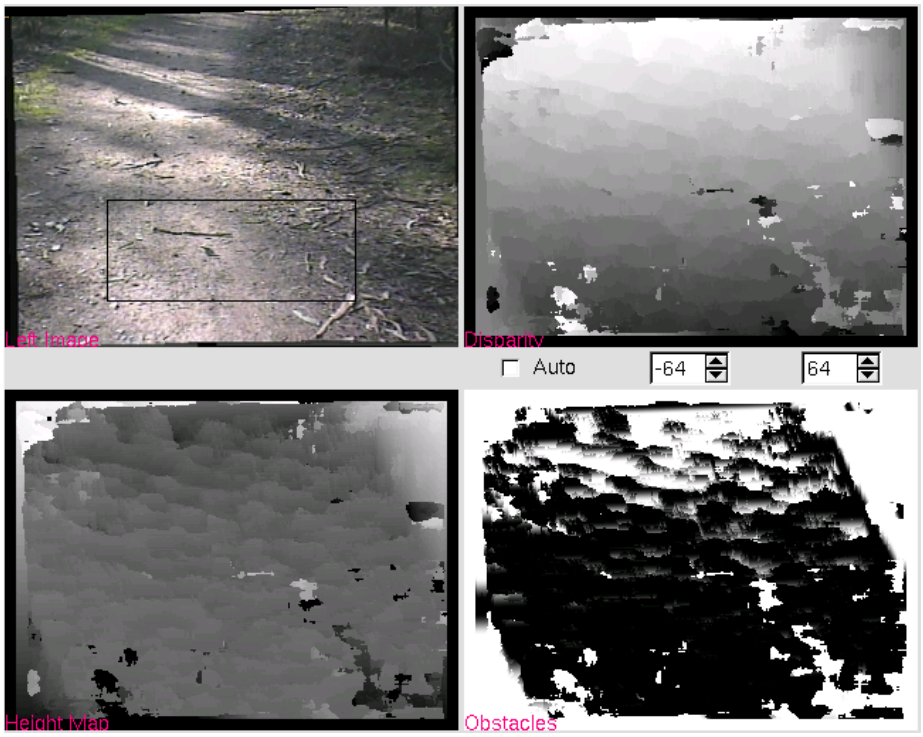


Figure 4.10: Clear space

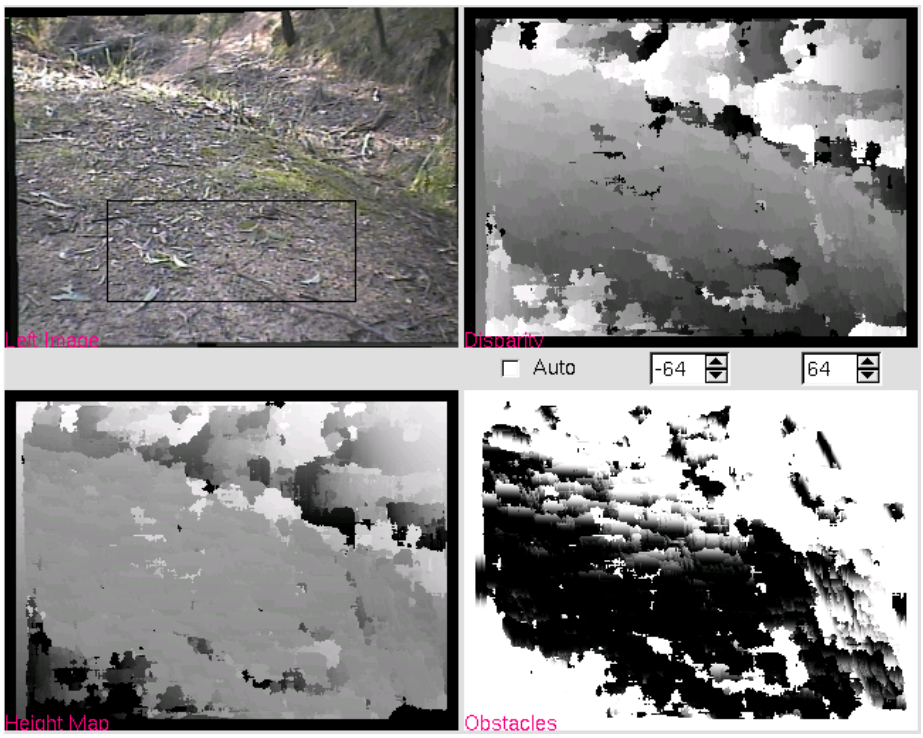


Figure 4.11: Aqueduct/gully

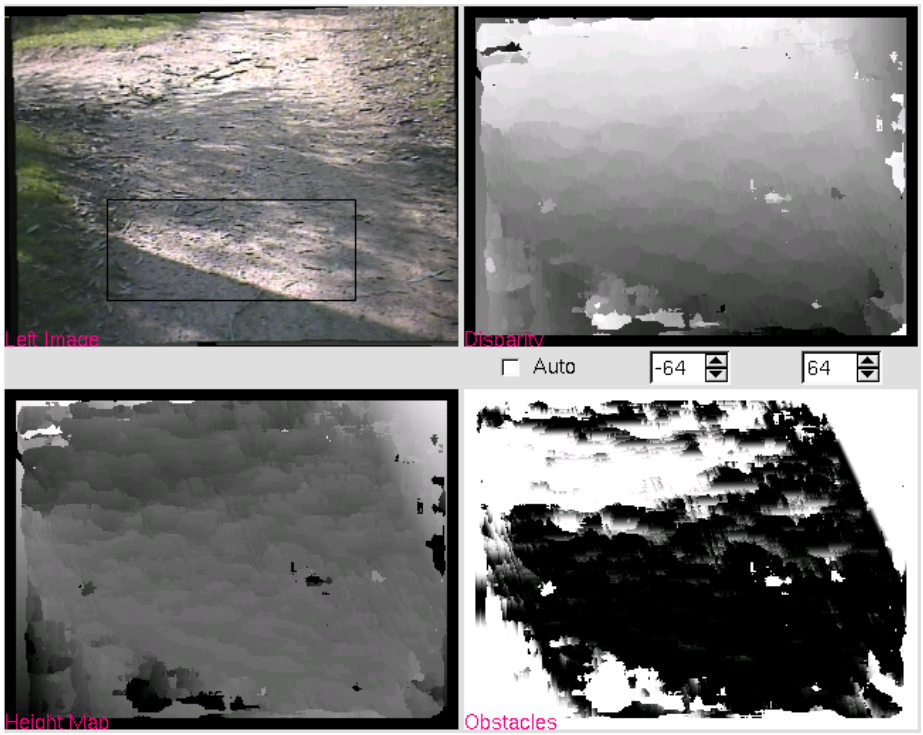


Figure 4.12: Sloping path to the left

terrain is safe except for this sloping region. While the unsafe region could be seen to extend further into the main path than necessary, this only serves to guide the robot to the right and away from the slope. Again, after progressing forwards, the robot would sense and adapt to the terrain, still seeing a drop to the left but a safer overall path forwards, as shown in Figure 4.13.

Figure 4.14 shows a region of ground that is unoccupied and hence safe except for a moderate-sized rock in the centre. (The shadow of a high-power transmission-line pylon is also clearly visible.) The rock, which is large enough to pose a problem (according to the divergence thresholds) is clearly shown to be an obstacle. Figure 4.15 shows the effect of varying the divergence ceiling. In Figure 4.6 d_{ceil} is reduced from 100 to 50, the effect of which is to show more of the rock as being an obstacle and also to introduce more spurious small regions misclassified as being obstacles. In Figure 4.6 d_{ceil} is increased to 200. This has the opposite effect of showing less of the rock as being an obstacle and also reducing the number of small areas that are misclassified. As demonstrated, a ceiling of approximately 100 is a good balance between suppressing some misclassified regions (those whose disparities and hence heights and divergences vary slightly from the true values) and correctly identifying true obstacles.

Figure 4.16 shows a near-invisible (with respect to a single, left-camera image) sudden drop in the terrain. The drop occurs approximately two-thirds of the way up the raw image and is barely visible as a slight inconsistency in ground textures and patterns. However, the stereo system clearly and correctly picks this up. The seeming "noise" in the top corners of the disparity map is most likely a result of those image regions being very far away (because

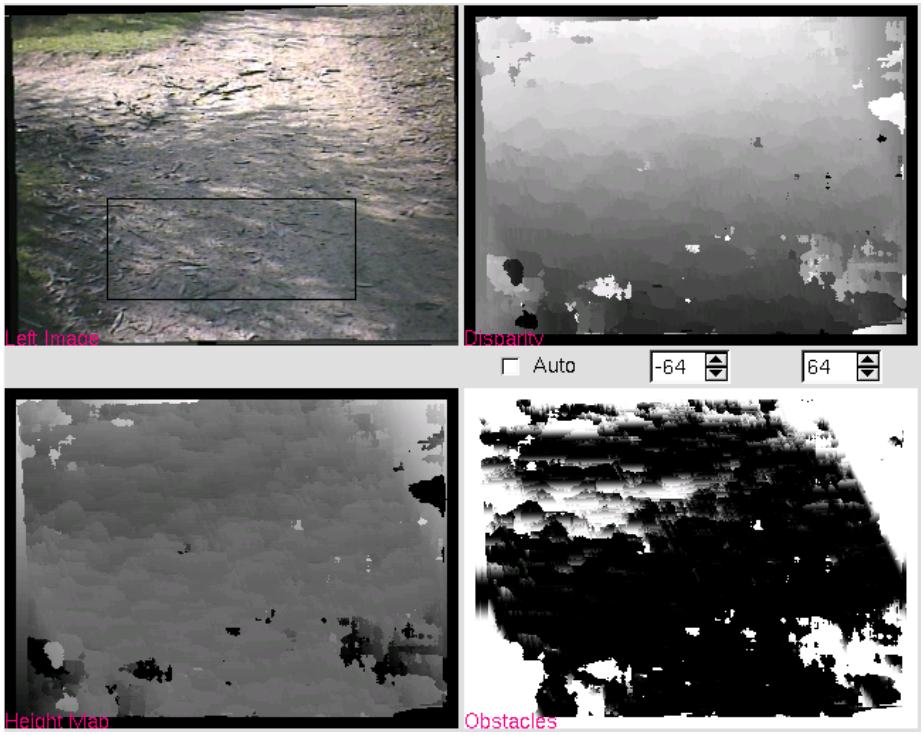


Figure 4.13: Sloping path to the left, closer

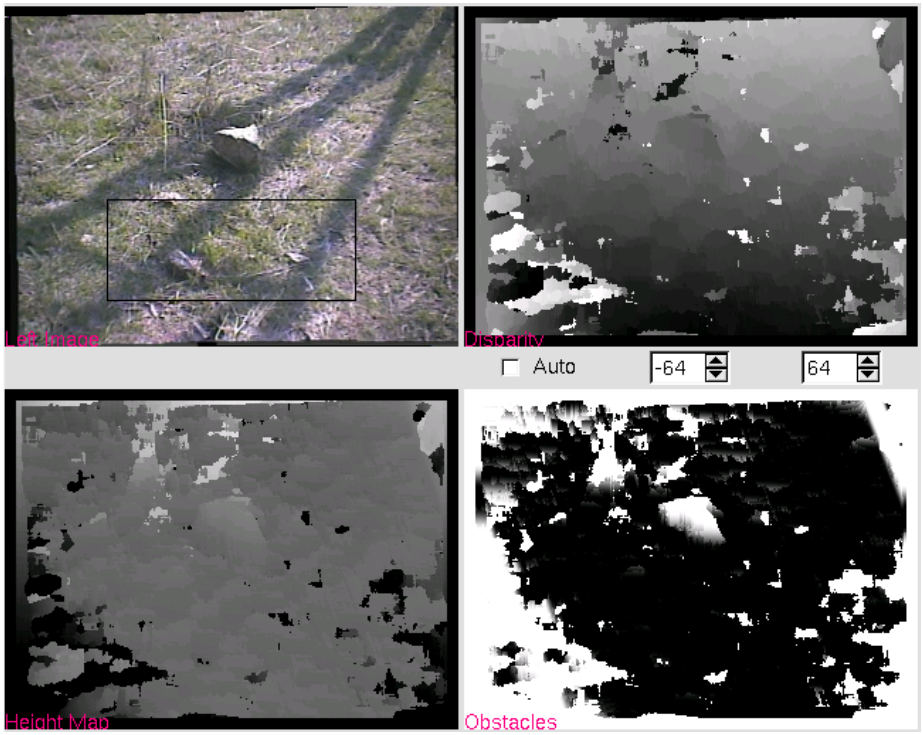
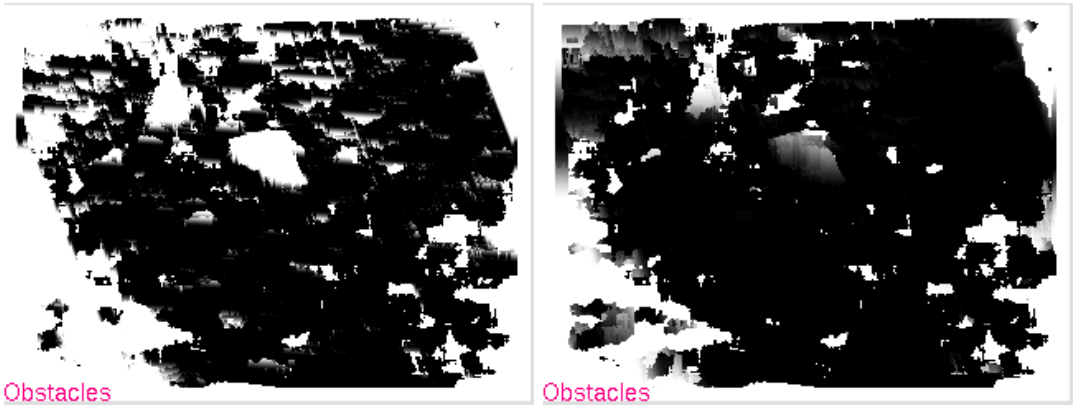


Figure 4.14: A medium-sized rock



(a) $d_{ceil} = 50$ (b) $d_{ceil} = 200$

Figure 4.15: A medium-sized rock, variation in divergence ceilings

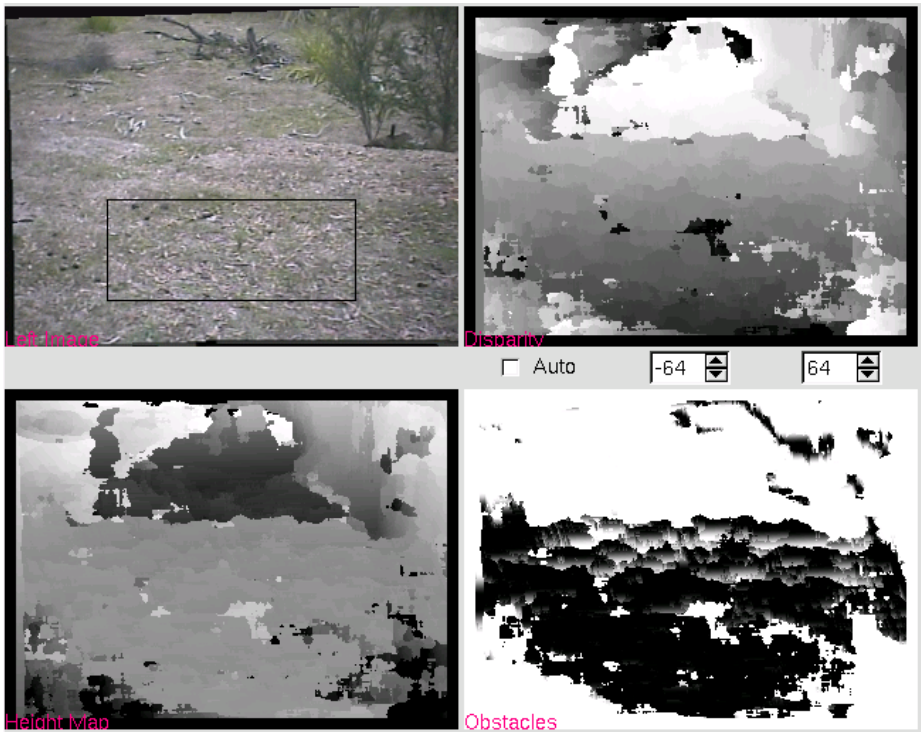


Figure 4.16: A sudden change in elevation

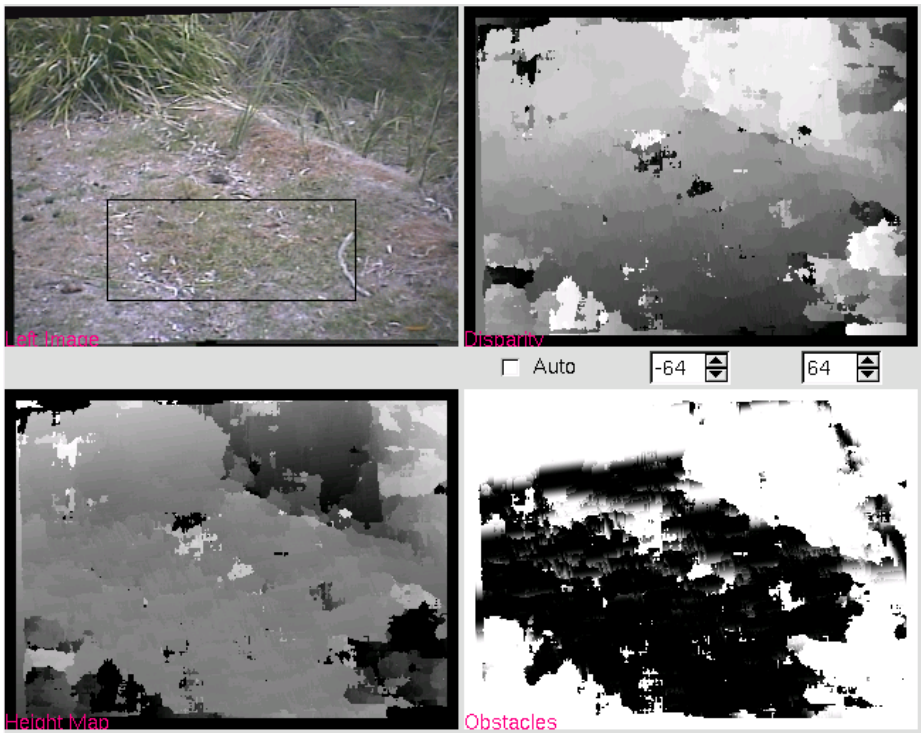


Figure 4.17: Positive and negative obstacles

of the drop) and hence there being no left-to-right image overlap (and so no correct matches). The resultant obstacle map clearly shown the correctly classified safe terrain in the foreground but a clear and large area of danger where the ground drops away.

The result shown in Figure 4.17 depicts a scene with both positive (above-ground) and negative (below-ground) obstacles. In the top-left, a large shrub blocks the way and is correctly shown as a region containing an obstacle. In the top-right, the ground drops away down a steep (though short) embankment. Again, this region is correctly classified as being unsafe, while the clear (and safe) foreground is correctly considered safe and essentially obstacle-free.

Of note is the common occurrence of obstacle regions in the lower-left and upper-right areas of the obstacle map, in spite of these areas often being safe (as seen in the raw images). This is the result of using a converging stereo rig with a wide baseline. With respect to the left camera, the lower-left and upper right regions will commonly have no overlap with the right camera (depending on the scene, these areas may but typically do not) and as such there are no correct matches for correspondence. The erroneous matches are then typically erroneous enough to result in them being considered unsafe. This fits in with the underlying pessimistic philosophy whereby, in the absence of data to the contrary, we assume the worst case.

A collection of additional experimental results are given on the accompanying CD-ROM, referred to in Appendix .6.

4.7 Conclusion

In this chapter, we presented one of the important problems faced by a mobile robot, especially one operating in bush environments: the problem of detecting obstacles. For a number of reasons, traditional obstacle detection systems using LIDAR or sonar scanners are not adequate, largely because they typically only provide a planar scan at best without the use of complexity-increasing pan-tilt units. Instead, an obstacle detection system based on passive stereo-vision was proposed that aimed to show the presence of anything that would serve to impede the motion of a robot. The result of the ideas and algorithms presented in this chapter is an obstacle likelihood map, a map that indicates for each point in the reference image (the left stereo image) the likelihood of that point being an obstacle, based on its divergence from a dynamically estimated ground plane. The use of a dynamically estimated ground plane provides a means by which the system can adapt to local changes in terrain that, while not unsafe, could easily be misclassified as potential dangers if a rigid and constant reference (that is, the calibrated ground plane) was used instead. However, to ensure only reasonable ground estimates are used, reference *is* made to the calibrated ground by way of a ground-normal threshold, whereby the estimated ground plane's normal must lie within a cone defined by the calibrated ground normal and a threshold angle. As demonstrated, the system handles both positive (above-ground) and negative (below-ground) obstacles equally well, and the divergence parameters can be used to tune the system with regards to what should be considered an obstacle. The system is also designed to be *fail-safe*: it errs on the side of caution and pessimistically assumes a region is unsafe unless the stereo data confirms otherwise. While potentially limiting in terms of optimality of resultant paths, the underlying motivation is safety and self-preservation above all.

Chapter 5

Detecting Poorly Structured Dirt Roads using Colour Vision

“ The old lost road through the woods,
But there is no road through the woods. ”
Rudyard Kipling, 1865–1936, “The Way through the Woods”

Figure 5.1: A typical, degraded dirt road

Navigation in bush or forest environments is a challenging task. These are largely natural environments, painted by the seemingly random strokes of nature’s hand. The underlying terrain follows no discernible pattern, with hills and valleys falling wherever they may. Most of the landscape is covered with vegetation, both small and large, whose appearance changes, either slowly with the passage of time or more rapidly due to extreme weather or other

environmental effects (an extreme case being the devastation caused by a bushfire). On the surface, mud, soil, clay, gravel and rocks appear intermingled and these too can change with the passage of time. Very little can be considered constant in this environment where irregularity and apparent randomness reign supreme.

5.1 Introduction

In spite of this high degree of apparent disorder, structure does exist in many areas in the form of unsealed dirt and gravel roads. These passageways have usually been put in place by park rangers, forest-management agencies and fire-fighting authorities to assist in their duties. They are rough and irregular and often poorly maintained, in no small part due to the typically small amount of traffic they see.

A typical example of a dirt road in a National Park is shown in Figure 5.1. In this example, vegetation has begun to form in patches along the centre of the road. This is a common occurrence on vehicular access roads: it is the disruption caused by tyres that keeps the road relatively clear.

Despite such ill-definition, these roads present a significant increase in structure when compared to the un-managed and typically tree-covered landscape. Planned navigation that utilises these roads will invariably be safer, faster, more efficient (for example, in terms of energy use) and more reliable than the brute-force alternative of cross-country trekking. Not all areas of interest may be able to be reached directly by dirt roads, but the probability of a successful mission can be increased by travelling along roads as much as possible.

These pathways are most often simply stretches that have been roughly cleared of trees, bushes, large rocks and similar debris, down to the underlying earth. Their surfaces are rough and bumpy and easily disturbed, their edges are irregular, unclear and vague at best, and the paths they take wind their way along as the surrounding landscape dictates. These roads, without asphalt, gutters or markings, are a far cry from those encountered in the suburbs or cities. An attempt to traverse them autonomously, therefore, requires an approach very different from that taken by traditional road-following techniques that primarily target urban streets.

Previous approaches to road-following have chiefly tackled sealed and paved urban roads (see Section 2.3.3 for more details). These roads provide a number of cues – such as their consistent geometry, painted markings and gutters – that allow many valid assumptions to be made, simplifying the task of finding the roads to begin with. These assumptions simply do not hold with the dirt roads present in bush environments, so an alternative approach much be used.

As mentioned, dirt roads in bush environments are most often passageways that have been roughly cleared of vegetation, typically to the extent that the underlying earth is exposed. Continued use (however infrequent), tends to mean this bare state is largely upheld. This observation leads to one of the few characteristics that can be utilised and taken advantage of: dirt roads look different from their surrounds. The difference in materials between the road – usually made up of a combination of dirt and gravel – and the surrounds – typically vegetation, provides a difference in appearance that is especially notable in terms of colour.

Accordingly, work focused on autonomous traversal of dirt roads typically involves the use of colour vision and various forms of region-classification. This is most commonly coupled with geometric road models that serve to specify which regions in an image contain the road and

hence should be sampled to characterise the road. Section 2.3.3 provides many more details of earlier work. The work presented in this chapter aims to provide a less restrictive road-finding system based on colour vision. By relaxing assumptions about road geometry and using colour-similarity-based likelihoods, we aim to produce a system that can successfully tackle even highly-degraded and poorly maintained dirt roads. Additionally, we aim to produce a system that is robust against adverse environmental conditions, most notably strong sunlight that produces harsh shadows that easily confuse many existing vision-based systems.

5.2 The Physical Setup

In order to find dirt roads using vision, there is only one simple requirement in terms of hardware: a camera. That used is a standard $\frac{1}{3}-$inch CCD video camera with a stated sensor resolution of 537×597 pixels, producing an analog PAL signal. It is housed in a rugged security-style metal case and uses CS-mount lenses. The lens used is of 6.0 mm focal length, giving an effective field-of-view of approximately 43 degrees horizontally and 33 degrees vertically. The camera is coupled to a standard PC frame-grabber, specifically a FlyVIDEO 3000 PCI capture card, which uses the Philips SAA7134 chipset. Images of 320×240 pixels are used.

The camera is mounted approximately one metre off the ground at the front of the robot, centred and facing forward. It is angled down to better view the road. To facilitate subsequent processing (whereby stereo results are combined with road-finding results), the field-of-view of the road-finding camera overlaps the region covered by the stereo pair (see Section 4.2 for details). This means there is a similar blind-spot: the ground directly in front of the robot up to a point approximately 1335 mm away is not visible. This is not seen as a serious problem, however, as the underlying philosophy is to look and plan ahead and only move into space that is considered safe and clear. Accordingly, what is invisible at any time *should* have been seen and processed (and declared part of a viable path) previously.

The overall camera placement on the robot is shown in Figure 5.2, with Figure 5.3 providing a more detailed view of the forward-facing camera rig. Also included in Figure 5.2 are the grids used during calibration. In Figure 5.4, the same calibration scene is shown from the perspective of the road-finding camera (Figure 5.4(a)) and the left stereo camera (Figure 5.4(b)), showing both the overlapping field-of-view and, through their absence compared to Figure 5.2, the blind-spots of each camera. As seen, the position and orientation of the road-finding camera is a balance between pointing down to see the immediate road and pointing up to see enough of the road ahead to perform useful planning.

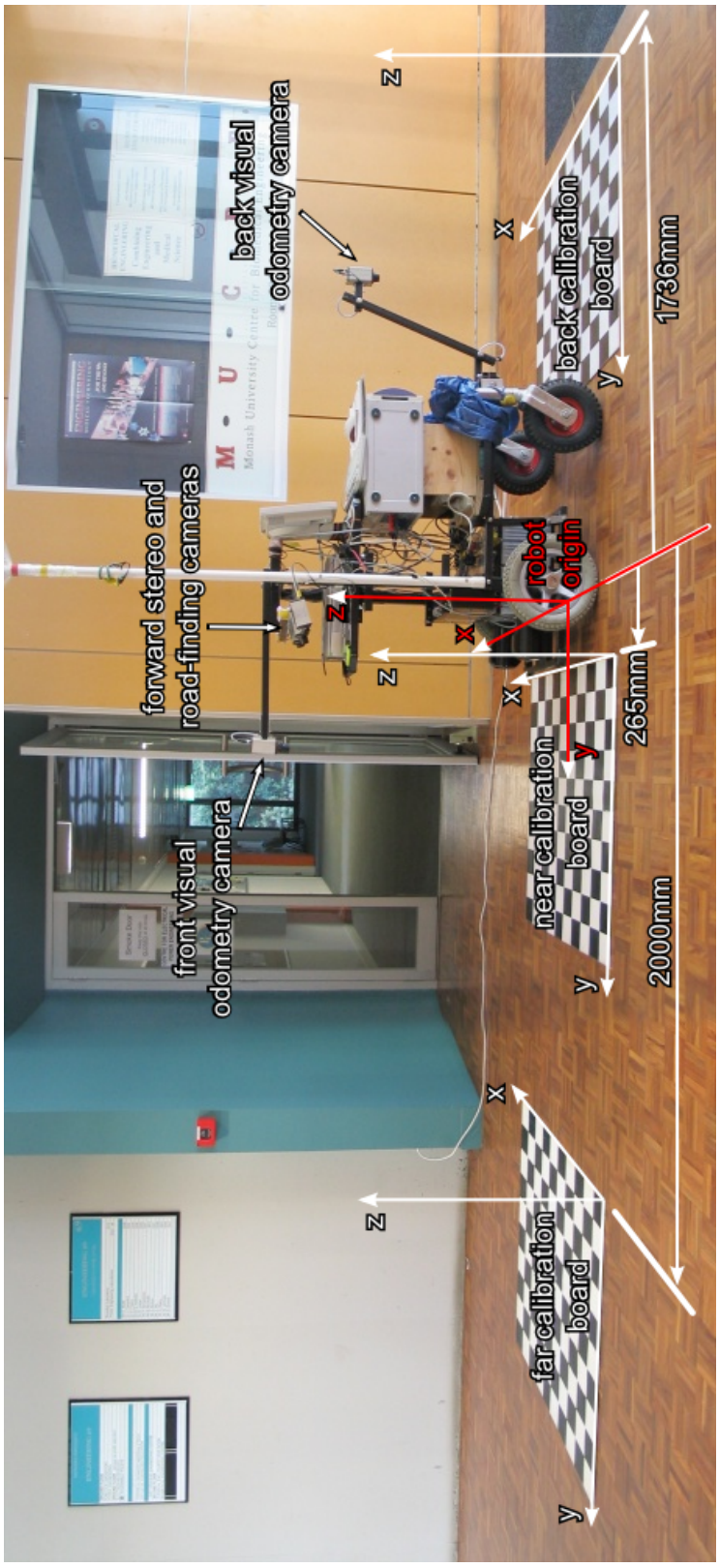


Figure 5.2: Overall camera setup with calibration grids

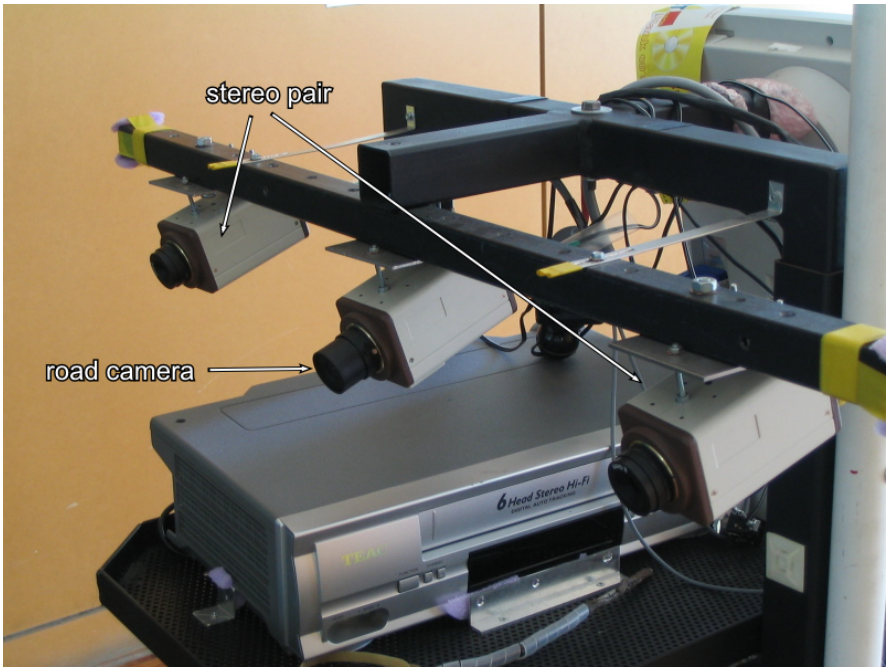
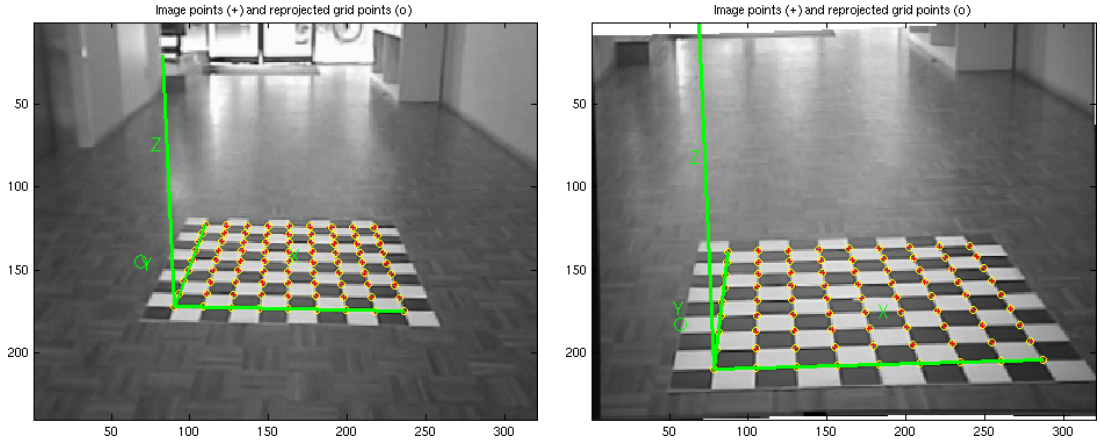


Figure 5.3: Forward-facing camera rig

5.3 Follow the Yellow-Brick Road

The central assumption of this system – and many other similar systems – is that the road to be followed can be described in some quantitative manner that is distinct and hence separable from the surrounding environment. Here we use pixel colour as the characterising parameter based on the observation and assumption that, typically due to a difference in material make-up, the road surface is of a different colour to the surrounding regions. Some previous systems have used off-line training [137, 114] to create descriptions of what are considered "on-road" regions. A potentially more flexible approach is one that samples the immediate environment, thereby adapting to the current conditions. This method has been used with some success previously [36, 43] and is the idea followed in the system presented here.

The problem can therefore be separated into two parts: defining the road and separating it from the surrounds. The first step takes as its input an image and aims to produce a parameter-set that defines the road as seen in that image. We aim to produce a statistical, colour-based description of the road surface, following the assumption that part of the image will always contain the road, and hence will provide an accurate sample of the road in the local region. The second step takes a collection of parameter-sets (more than one set is used to provide temporal continuity) and determines the likelihood that any image region belongs to the road, based on the provided parametric description. The likelihood image produced will then be used by further processing stages to determine a local path.



(a) View from road camera (b) View from left stereo camera

Figure 5.4: Two views of the world

5.4 Defining the Dirt

In order to define the dirt road in a manner that allows it to be extracted and separated from its surrounds, certain assumptions need to be made. The prime assumption being made here is that the road surface displays colour-space statistics that are different from those of the surrounding regions. This allows an image of the road to be partitioned by filtering it into pixels that potentially belong to the road (they are a similar colour) and those that do not (they are different enough to be ignored). To allow for uncertainty, the distinction between road and not-road is not abrupt: pixels are given a likelihood scoring based on how similar they are to the road colour model.

5.4.1 Representing Colour

In much the same way that a three-dimensional point in space can be represented in a number of different coordinate systems (such as Euclidean, Cylindrical and Spherical), the colour of a pixel in an image can be represented in a number of ways. The choice of representation often has a strong bearing on the efficiency and complexity of a solution to a given problem. Digital colour images are almost always captured in what can be viewed as a Euclidean colour space known as RGB (Red, Green, Blue), where the three rectangularly-orthogonal axes represent the amount of red, green and blue in a given colour (following a light-based, additive colour process, as distinct from a pigment-based, subtractive process that typically has axes of cyan, magenta and yellow).

One of the biggest problems faced by image-processing algorithms outdoors is that of lighting. Unlike indoors, lighting cannot be controlled, nor can it be adequately predicted. A scene that is uniformly lit due to overcast cloud cover can quickly become full of strong sunlight and shadows when the clouds move. In the context of road-finding, we wish to make our system as robust to these sorts of environmental changes as we can. One way to do this is to use a colour representation that separates overall lighting intensity which can then be treated with greater tolerance.

A number of colour models exist that have this property, with one of the most commonly used being HSI (Hue, Saturation, Intensity) and its variants (including HSV (Hue, Saturation, Value) and HSL (Hue, Saturation, Lightness)). These are all cylindrical models, with hue (a measure of colour, such as red, orange, green and so on) being an angle about an intensity axis (or value or lightness axis), and saturation (the "colourfulness", ranging from no colour (greyscale) to vibrantly colourful) being a radial distance from the intensity axis. A problem with these representations shows itself when we try to perform statistical analysis on data with these representations. The circular nature of hue (as hue is an angle and hence repeats every 360 degrees or 2π radians), makes certain calculations (such as means and variances) overly complex, without offering any real benefits.

In addition to these models, there exist the commonly-used rg-chromaticity and YCbCr models that also serve to separate intensity information (in the case of rg-chromaticity excluding it altogether). Either of these could potentially be used (rg-chromaticity can be augmented with the lost intensity information), though neither present strong arguments for their use.

Another point worth noting here is that many of the existing colour models (and almost all of those mentioned above) have been created for purposes other than image-processing (rg-chromaticity being the exception). HSI, HSV and HSL, for example, have their primary utility in allowing colours to be described and chosen by *people* in an intuitive manner [112]. YCbCr and related models on the other hand have their origins in the transmission of images (largely video sequences) and are geared towards efficient use of available bandwidth, again taking into consideration human perception [111]. Accordingly, without offering any great advantages, there is no good reason to choose any of these representations.

Instead, to maintain simplicity, we will make use of a colour model, which we denote as a XYI (X, Y, Intensity), that is a simple linear transformation of an RGB-represented colour, and that shares the decoupled intensity axis of HSI but uses a pair of rectangularly-orthogonal axes to represent the "colour" information, thereby providing the desired intensity separation while allowing simple computation.

The conversion from RGB to XYI can be achieved through a simple coordinate transform consisting of a rotation about a specified axis to produce a set of rectangular axes, one of which is aligned with the line $R = G = B$. Specifically, we can rotate an RGB-tuple $\boldsymbol{p}_{RGB} = [r, g, b]^T$ about an axis $\boldsymbol{\alpha} = [\frac{-1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0]^T$ by an angle $\theta = \text{Tan}^{-1}(\sqrt{2})$ to produce the desired XYI-tuple $\boldsymbol{p}_{XYI} = [x, y, i]^T$. We use the following formula for an angle-axis rotation:

$$\boldsymbol{p}_{XYI} = \mathsf{R}(\boldsymbol{\alpha}, \theta)\boldsymbol{p}_{RGB} \tag{5.1}$$

where

$$\mathsf{R}(\boldsymbol{\alpha}, \theta) = \begin{bmatrix} \alpha_x^2 + (1 - \alpha_x^2)C_\theta & \alpha_x\alpha_y(1 - C_\theta) + \alpha_z S_\theta & \alpha_z\alpha_x(1 - C_\theta) - \alpha_y S_\theta \\ \alpha_x\alpha_y(1 - C_\theta) - \alpha_z S_\theta & \alpha_y^2 + (1 - \alpha_y^2)C_\theta & \alpha_z\alpha_y(1 - C_\theta) + \alpha_x S_\theta \\ \alpha_z\alpha_x(1 - C_\theta) + \alpha_y S_\theta & \alpha_z\alpha_y(1 - C_\theta) - \alpha_x S_\theta & \alpha_z^2 + (1 - \alpha_z^2)C_\theta \end{bmatrix} \tag{5.2}$$

and $C_\theta = \cos(\theta)$ and $S_\theta = \sin(\theta)$ for brevity.

Figure 5.5: The road sampling window

For $\boldsymbol{\alpha} = [\frac{-1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0]^T$ and $\theta = \text{Tan}^{-1}(\sqrt{2})$ we then have

$$\begin{bmatrix} x \\ y \\ i \end{bmatrix} = \begin{bmatrix} \frac{1}{6}(3+\sqrt{3}) & -\frac{1}{6}(3-\sqrt{3}) & -\frac{1}{\sqrt{3}} \\ -\frac{1}{6}(3-\sqrt{3}) & \frac{1}{6}(3+\sqrt{3}) & -\frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} \tag{5.3}$$

It should also be noted that conventional normalisation is performed to give in inputs $r, g, b \in [0, 1]$ and transformed values $x, y, i \in [0, 1]$.

5.4.2 Obtaining a Statistical Description

In order to find the road in an image we wish to extract a concise description of the road in terms of its colour. We have one geometric constraint that is represented by a sampling window within the image (shown in Figure 5.5), following the assumptions that this window contains predominantly road surface and that these pixels form an adequate sample of the local road surface.

A common technique for characterising images or image regions is the use of colour histograms. Firstly a histogram is created from a set of samples (the characterising region) that describes the number of samples of each colour. To determine the similarity of another test region to the reference region, a histogram of the test region is created and compared against the reference histogram, with the similarity of histograms being a measure of the similarity of the regions, in terms of the distribution of colours. For a number of reasons, colour histograms are not considered to be adequate for the current task. Firstly, a histogram is a low-level description, and as such high-level adjustments, such as applying a greater tolerance to variation in intensity, are more difficult to perform. Secondly, a histogram is only valid for an image *region*, meaning we must test image *regions*, and not individual pixels, against a reference histogram, typically leading to a sparse similarity map unless, for example, overlap-

ping regions are used. Finally, for a reasonably large number of histogram bins – that is, for a reasonably high colour resolution – histogram matching can be slow, especially if a dense similarity map is to be produced.

A simple and effective alternative is to calculate the colour statistics (specifically the mean and standard deviation) of the pixels within the sample window. Very simple distance measures then allow a single pixel to be compared against the statistics, leading to the fast calculation of a dense similarity map while maintaining high colour resolution.

To accommodate variations in lighting, a pixel representation was chosen that allows light intensity to be made effectively independent of colour. Further, to allow for simple statistical analysis, the colour space avoids the use of cylindrical coordinates (a feature of the more common intensity-invariant colour spaces). We aim to produce a descriptor of the road $\mathbf{D}_R$ containing the mean $\bar{c}$ and standard deviation S of the pixel colours within a defining region. As mentioned many times previously, there exists a great deal of irregularity and visual richness in the bush environment. In the context of finding dirt roads using colour vision, this results in the need for a flexible approach to road characterisation. Our aim is to describe the road surface appearance (specifically the colour of the road) in order to determine where the road is. Unlike in urban environments with sealed roads, dirt roads in bush environments more often than not consist of a variety of materials with different appearances. Soil is brown, clay is red, sand is yellow, and gravel can be a mix of grey, beige, yellow and brown. Further to this, shadowed areas are dark while well-lit regions are bright. In the space of a few metres of road, it is not at all uncommon to encounter all of these visually different features. To accommodate such changes, a single characterisation of the road is unlikely to be successful without being overly complex. Therefore, we opt to use multiple, simple, statistical characterisations that can allow for local variations.

To perform this task, we utilise the k-means algorithm [75], which allows us to cluster the pixels into a predefined number of groups. In the presence of multiple road materials, we are much better able to produce an accurate description of the road's appearance. The k-means algorithm's use of a fixed (and hence pre-determined) number of clusters *could* be seen as a disadvantage when compared to clustering algorithms that use a variable number of clusters, such as the popular QT-Clust algorithm [53]. However, this allows the potential number of filters (and hence the computation required in further processing stages) to be controlled and limited easily. Furthermore, the k-means algorithm is significantly less computationally expensive than QT-Clust, which is of particular importance if the road-finding system is to run at a reasonable speed.

Each cluster is defined by its centre $\bar{c} = [\bar{c}_x, \bar{c}_y, \bar{c}_i]$, equal to the mean of member colours, its spread $S = [S_x, S_y, S_i]$, equal to the standard deviation of member colours, and a member count N. The algorithm (shown in Algorithm 5.1) is made stable with regards to data that inherently has fewer clusters than the number sought by allowing clusters to have no members in the final result. However, to prevent clusters being starved due to poor initialisation (clusters are initialised with a random selection of data), during the processing loop clusters with no members are reset and given a zero standard deviation and a centre equal to the mean colour of the colour-space, as distinct from the mean colour of the data. For example, if each colour channel is an integer value in the range $[0, 255]$, the reset cluster will be given a centre of $\bar{c} = [127, 127, 127]$. A bound is also placed on the number of iterations allowable, in order to limit the time taken to perform the clustering. This follows a "*best effort*" philosophy, that is, obtaining the best result in the time allowed.

The resulting road descriptor $\mathbf{D}_R$ (shown in Figure 5.6) contains the mean $\bar{c}$ and standard

Algorithm 5.1: k-means parametrisation

Procedure: D = `Cluster(` I_{sample}, N_{sample}, K `)`

Input: image region to be sampled I_{sample} containing N_{sample} pixels
Input: requested number of clusters K

Output: set of $\hat{K}$ road descriptors, $D = \{\mathbf{D}_{R0}, \ldots, \mathbf{D}_{R\hat{K}-1}\}$

Data: iteration, maximum iterations, i, i_{max}
Data: reassignments, minimum reassignments, r, r_{min}
Data: intermediate statistical data,
$data_S = \{(sum_0, sqr_0, N_0), \ldots, (sum_{K-1}, sqr_{K-1}, N_{K-1})\}$
Data: intermediate pixel data (pixel colour and associated cluster),
$data_P = \{(c_0, K_0), \ldots, (c_{N_{sample}-1}, K_{N_{sample}-1})\}$

1: $data_P \leftarrow \emptyset$

▷ `associate pixels with an initially random cluster`
2: **for** $p \in I_{sample}$ **do**
3: $data_P \overset{\cup}{\leftarrow} (p,$ `Random` $(\ 0,\ K-1\ )\ )$
4: **end**
5: $i \leftarrow 0$ ▷ `initialise iterations`
6: $r \leftarrow \infty$ ▷ `initialise reassignments`
7: **while** $(i < i_{max}) \wedge (r > r_{min})$ **do**
8: **for** $k \in \{0, \ldots, K-1\}$ **do**
9: $data_S[k] \leftarrow (0, 0, 0)$ ▷ `reset cluster statistics`
10: **end**
▷ `recalculate cluster statistics`
11: **for** $p \in data_P$ **do**
12: $k \leftarrow p.K$ ▷ `get associated cluster`
13: $data_S[k].sum \overset{+}{\leftarrow} p.c$ ▷ `add to cluster's sum`
14: $data_S[k].sqr \overset{+}{\leftarrow} (p.c)^2$ ▷ `add to cluster's sum of squares`
15: $data_S[k].N \overset{+}{\leftarrow} 1$ ▷ `increment samples in this cluster`
16: **end**
17: **for** $k \in \{0, \ldots, K-1\}$ **do**
18: $N \leftarrow data_S[k].N$
19: **if** $N > 0$ **then** ▷ `recalculate cluster if it has some samples`
20: $S[k].\bar{c} \leftarrow data_S[k].sum/N$
21: $S[k].S \leftarrow \sqrt{data_S[k].sqr/N - S[k].\bar{c}^2}$
22: $S[k].N \leftarrow N$
23: **else** ▷ `otherwise seed cluster with mean colour`
24: $S[k] \leftarrow (\widehat{colour}, 0, 0)$
25: **end**
26: **continued ...**

Algorithm 5.1: continued

```
27: ... continued
      ▷ reassign pixels based on new clusters
28:   r ← 0                                          ▷ initially no reassignments
29:   for p ∈ data_P do
30:     δ_best ← ∞
31:     L_best ← 0
        ▷ determine the cluster nearest this sample
32:     for k ∈ {0, ..., K − 1} do
33:       δ ← ‖p.c − S[k].c̄‖
34:       if δ < δ_best then
35:         δ_best ← δ
36:         K_best ← k
37:       end
38:     end
39:     if K_best ≠ p.K then
40:       p.K ← K_best                                 ▷ assign new cluster
41:       r +← 1                                       ▷ increment reassignments
42:     end
43:   end
44:   i +← 1                                           ▷ increment iterations
45: end
    ▷ either the clusters have stabilised (reassignments below the
      threshold) or maximum number of iterations has been exceeded
```

deviation S of the estimated cluster, along with the number of samples N used to obtain these statistics and a measure of confidence that this descriptor is an accurate representation of the road, ζ. The road-sampling window contains N_{total} samples (pixels) that are distributed amongst $\hat{K}$ clusters (the number of clusters actually found and which is equal to or less than the K clusters requested). With a uniform distribution of samples, we would expect a quota of $\frac{N_{total}}{\hat{K}}$ samples to be assigned to each cluster. The confidence measure is the ratio of samples actually assigned to a cluster against this "uniform-assignment" quota, and is defined as follows:

$$\zeta = \frac{N \cdot \hat{K}}{N_{total}} \tag{5.4}$$

It is a description of how much data was used to produce a descriptor compared to an even spread of samples.

Road descriptor: $\mathbf{D}_R$	
mean	$\bar{c} = [\bar{c}_x, \bar{c}_y, \bar{c}_i]$
spread	$S = [S_x, S_y, S_i]$
members	N
confidence	ζ

Figure 5.6: Road-finding road colour descriptor

5.5 Separating the Surrounds

5.5.1 Colour Filtration

Each descriptor (defined as above) is used to create a colour filter, against which each image pixel is compared to determine the likelihood of it belonging to the road. Each filter represents an ellipsoid in XYI colour coordinates, centred on the filter's mean. The standard deviations of each colour component are used to determine the axes of the ellipse, each scaled by global (that is, filter-independent) tolerances.

Filter: **F**	
road descriptor	$\mathbf{D}_R$
age	t
total hits	h_{total}

Figure 5.7: Road finding colour filter

The filter data structure is shown in Figure 5.7. It is primarily defined by a road descriptor (detailed in Section 5.4.2) whose mean $\bar{c} = [\bar{c}_x, \bar{c}_y, \bar{c}_i]$ and standard deviation $S = [S_x, S_y, S_i]$ describe the location and spread of the filter in XYI colour coordinates and are analogous to the resonant frequency and bandwidth, respectively, of a traditional bandpass filter. Additionally, the filter's age and the total number of hits are stored to allow a calculation of the filter's utility. The filter's age is simply the number of processing iterations that it has been used, also equal to the number of processing iterations since its creation. The total hits parameter and the way it is used will be described in Sections 5.5.2 and 5.6.

For a given pixel of colour $[x, y, i]$, we define its score ρ against a filter using tolerances of $\Delta = [\Delta_x, \Delta_y, \Delta_i]$ as follows:

$$d = \sqrt{\left(\frac{x - \bar{c}_x}{\Delta_x S_x}\right)^2 + \left(\frac{y - \bar{c}_y}{\Delta_y S_y}\right)^2 + \left(\frac{i - \bar{c}_i}{\Delta_i S_i}\right)^2} \tag{5.5}$$

and

$$\rho = \begin{cases} minimum(\,1,\ (1-d)\cdot\zeta\,) & , d \leq 1 \\ 0 & , otherwise \end{cases} \tag{5.6}$$

where ζ is the road descriptor confidence, defined in Eq^n 5.4 and used as a weighting factor. Note also that the score ρ is limited to the range $[0.0, 1.0]$.

Typical tolerances, empirically determined, are $\Delta_x = 3.0$, $\Delta_y = 3.0$ and $\Delta_i = 4.0$, reflecting the desired greater tolerance to variations in intensity compared with variations in colour. We now have a way of determining the likelihood of a pixel belonging to the road, based on its similarity to a sampled road section.

In order to be efficient, the colour filters are implemented as lookup tables. This is feasible largely because of the choice of a sixteen-bit colour encoding scheme (each pixel has five bits for red, six bits for green and five bits for blue, known as RGB565) meaning the lookup table has a manageable size of 65,536 entries. The next higher standard pixel format would use twenty-four bits per pixel, requiring a table with 16,777,216 entries, 256 times that of the chosen format. Using sixteen bits per pixel, we have a good balance of resolution and manageable amounts of data.

5.5.2 Voting for the Road

Each colour filter produces a score in the range $[0.0, 1.0]$ for a given input colour pixel, indicating how well that pixel matches the filter. A score of 0 indicates that the pixel falls outside the bounds of the filter and a score of 1 indicates that the pixel falls well within the bounds of the filter. (Note that, because of the scaling by the descriptor confidence ζ in Eq^n 5.6, a maximum score is possible for any pixel falling within the filter's bounds, though is most likely to occur with pixels near the centre of the filter's space.)

With the system using a set of N_F filters, each pixel in the input road image produces N_F scores. Each score is already weighted by descriptor-associated confidence and, as there is no logical way to combine scores (a pixel could be given relatively high scores by a number of filters, but that is not necessarily an indication that it is more likely to belong to the road, just that the filters had an overlapping region), we choose to simply assign to each pixel the highest score it receives from tests against the N_F filters.

When the final score is assigned to a pixel, the owning filter (that is, the filter that gave the final score) has its total hit count (shown in Figure 5.7) incremented, indicating that that filter has produced another best match.

5.5.3 Efficient Filtering through Recursive Subdivision

As with many, if not most computer-vision tasks, methods are sought to improve processing performance, very often in terms of speed and efficiency. In the context of finding roads, we make the observation that, typically, the road lies within large and contiguous regions of an

image. It therefore makes sense to use a processing mechanism that exploits this in order to reduce the amount of computation required.

To do this, we make the assumption that regions that belong to the road appear clustered together. Beginning with a region that includes the entire image, the region is divided into N_{div} sub-regions. Each sub-region is then processed with granularity g, that is, every g^{th} pixel (both horizontally and vertically) is given a score ρ (given by Eq^n 5.6) by each filter, and the best score ρ_{best} is stored. If the best score is above a threshold, indicating an acceptable match, the sub-region is itself divided into N_{div} sub-regions, and each new, smaller region is processed at a finer granularity of $\frac{g}{2}$. This process repeats until either the granularity is one (that is, each pixel in the region is being processed) or the dimensions of a new sub-region are below a threshold, in which case the new sub-region has every pixel processed.

The process is depicted in Figure 5.8, where 1 is the initial area, subdivided into four regions. Upon finding a match within sub-region 2 (within the shaded area), 2 is sub-divided. Similarly, upon finding a match in sub-region 3, 3 is subdivided. A match found in sub-region 4 results in a complete processing of that area, as it is below the minimum sub-region size. If no matches are made within a region, no further subdivision or processing is performed on that region.

The aim is to trade off some of the speed increase of a simple blanket sub-sampling for increased detail. The usefulness of this method is based upon the premise that the pixels of interest, that is, those that will be passed by the filter, are typically found clustered together and separated by regions of no interest. As a result, large areas of the image that contain no (or very few) interesting pixels can effectively be skimmed over and processed very quickly. For the majority of images to be encountered, it is believed that the speed-up of reduced processing will far outweigh the overheads associated with subdivision and recursion. It should be noted, however, that by its very nature this method has a processing time that is highly dependent upon the image being processed. Additionally, because of the quantised (granular) processing, it is possible that some pixels of interest may be overlooked, particularly those that are isolated or in a very small group. However, this is not seen as detrimental, given the underlying desire to find *regions* of interesting pixels.

The algorithms used to perform this filtering operation are shown in Algorithm 5.2, Algorithm 5.3, Algorithm 5.4 and Algorithm 5.5. The top-level procedure (Algorithm 5.2) is almost trivial and simply serves to initialise the score-image and trigger the first subdivision. Next, in Algorithm 5.3, we selectively subdivide the supplied region, subject to it being larger than the minimum allowable region and the requested granularity being greater than the minimum allowable. If either the region is too small or the granularity too fine, we simple process every pixel at the minimum granularity, via Algorithm 5.4. Otherwise, each newly created sub-region is processed recursively using Algorithm 5.5. Both of these procedures simply scan through the pixels in the given region, testing each against each of the filters. They differ in what is done when a pixel is found have a high enough score. In Algorithm 5.4 we simply store the best score in the output image and continue processing the current region. In Algorithm 5.5, upon finding an acceptable match, we stop processing and instead further subdivide the current region.

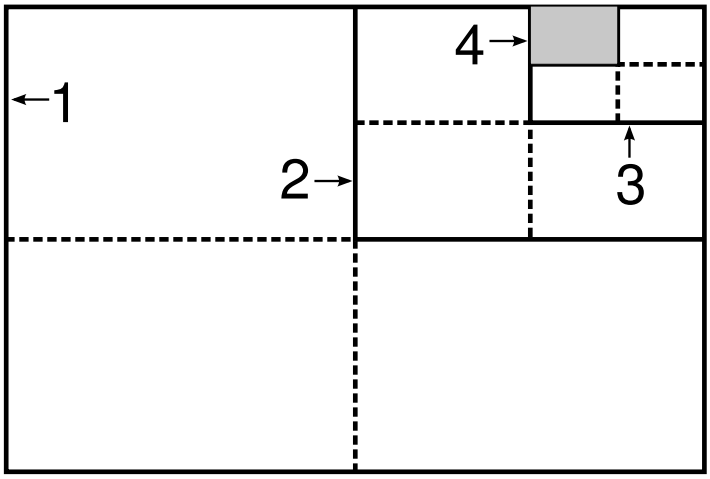


Figure 5.8: Recursive subdivision for processing

Algorithm 5.2: Top-level filtering

Procedure: S = `FilterImage(` I, F, x_0, y_0, x_1, y_1, ρ_{min} `)`

Input: input image I
Input: filter set F
Input: processing region $(x_0, y_0) \rightarrow (x_1, y_1)$
Input: minimum allowable score ρ_{min}

Output: output road-score image S

Data: initial (coarsest) processing granularity, g_{start}

1: $S \leftarrow \emptyset$ ▷ `initialise scores to zero`

▷ `perform first subdivision`
2: $S \leftarrow$`SubDivide (` I, F, x_0, y_0, x_1, y_1, g_{start}, ρ_{min} `)`

Algorithm 5.3: Image subdivision

Procedure: S = `SubDivide(` I, F, x_0, y_0, x_1, y_1, g, ρ_{min} `)`

Input: input image I
Input: filter set F
Input: processing region $(x_0, y_0) \rightarrow (x_1, y_1)$
Input: processing granularity g
Input: minimum allowable score ρ_{min}

Output: output road-score image S

Data: minimum sub-region size W_{min}, H_{min}
Data: minimum granularity g_{min}
Data: number of sub-regions N_{div}

▷ `determine region dimensions`
1: $W_{reg} = \boldsymbol{p}_0.x - \boldsymbol{p}_1.x$
2: $H_{reg} = \boldsymbol{p}_0.y - \boldsymbol{p}_1.y$
▷ `determine if region is too small or granularity is too fine`
3: **if** $(W_{reg} \leq W_{min}) \vee (H_{reg} \leq H_{min}) \vee (g_{new} \leq g_{min})$ **then**
▷ `if so, process the entire region at the finest granularity`
4: $S \leftarrow$`FilterStandard` (I, F, x_0, y_0, x_1, y_1, g_{min}, ρ_{min})
5: **else**
▷ `otherwise, subdivide`
6: $g_{new} = g/2$
7: $W_{reg} \leftarrow W_{reg}/N_{div}$
8: $H_{reg} \leftarrow H_{reg}/N_{div}$
9: **for** $j \in \{0, \ldots, N_{div} - 1\}$ **do**
10: **for** $i \in \{0, \ldots, N_{div} - 1\}$ **do**
▷ `determine limits of new sub-division`
11: $x'_0 \leftarrow x_0 + i \cdot W_{reg}$
12: $x'_1 \leftarrow x'_0 + W_{reg}$
13: $y'_0 \leftarrow y_0 + j \cdot H_{reg}$
14: $y'_1 \leftarrow y'_0 + H_{reg}$
▷ `process new sub-division`
15: $S \leftarrow$`FilterRecursive` (I, F, x'_0, y'_0, x'_1, y'_1, g_{new}, ρ_{min})
16: **end**
17: **end**
18: **end**

Algorithm 5.4: Pixel Filtering, Standard

Procedure: S = `FilterStandard(` I, F, x_0, y_0, x_1, y_1, g, ρ_{min} `)`

Input: input image I
Input: filter set F
Input: processing region $(x_0, y_0) \rightarrow (x_1, y_1)$
Input: processing granularity g
Input: minimum allowable score ρ_{min}

Output: output road-score image S

Data: coordinates and pixel to be tested $p \leftarrow I[y, x]$
Data: best score ρ_{best}

1: $y \leftarrow y_0$
2: **while** $y < y_1$ **do**
3: $x \leftarrow x_0$
4: **while** $x < x_1$ **do**
5: $p \leftarrow I[y, x]$ ▷ `get next pixel to process`
6: $\rho_{best} \leftarrow \rho_{min}$ ▷ `initial best score`
7: **for** $f \in \{0, \ldots, N_F - 1\}$ **do**
▷ `test pixel` p `against the` f^{th} `filter`
8: $\rho \leftarrow$ `FilterTest` $(\ F[f],\ p\ )$
9: **if** $\rho > \rho_{best}$ **then** ▷ `check if we have a better score`
10: $\rho_{best} \leftarrow \rho$
11: **end**
12: **end**
13: **if** $\rho_{best} > \rho_{min}$ **then**
14: $S[y, x] \leftarrow \rho_{best}$ ▷ `store the best score`
15: **else**
16: $S[y, x] \leftarrow 0$ ▷ `store a zero score`
17: **end**
18: $x \overset{+}{\leftarrow} g$
19: **end**
20: $y \overset{+}{\leftarrow} g$
21: **end**

Algorithm 5.5: Pixel Filtering, Standard

Procedure: S = `FilterRecursive(` I, F, x_0, y_0, x_1, y_1, g, ρ_{min} `)`

Input: input image I
Input: filter set F
Input: processing region $(x_0, y_0) \rightarrow (x_1, y_1)$
Input: processing granularity g
Input: minimum allowable score ρ_{min}

Output: output road-score image S

Data: coordinates and pixel to be tested $p \leftarrow I[y, x]$
Data: best score ρ_{best}

1: $y \leftarrow y_0$
2: **while** $y < y_1$ **do**
3: $x \leftarrow x_0$
4: **while** $x < x_1$ **do**
5: $p \leftarrow I[y, x]$ ▷ `get next pixel to process`
6: $\rho_{best} \leftarrow \rho_{min}$ ▷ `initial best score`
7: **for** $f \in \{0, \ldots, N_F - 1\}$ **do**
▷ `test pixel` p `against the` f^{th} `filter`
8: $s \leftarrow$ `FilterTest` ($F[f]$, p)
9: **if** $\rho > \rho_{best}$ **then** ▷ `check if we have a better score`
10: $\rho_{best} \leftarrow \rho$
11: **end**
12: **end**
13: **if** $\rho_{best} > \rho_{min}$ **then** ▷ `determine whether we have found a match`
▷ `we have a match, so subdivide this region`
14: $S \leftarrow$`SubDivide` (I, F, x_0, y_0, x_1, y_1, g_{start}, ρ_{min})
15: $x \leftarrow x_1$;
16: $y \leftarrow y_1$;
17: **end**
18: $x \overset{+}{\leftarrow} g$
19: **end**
20: $y \overset{+}{\leftarrow} g$
21: **end**

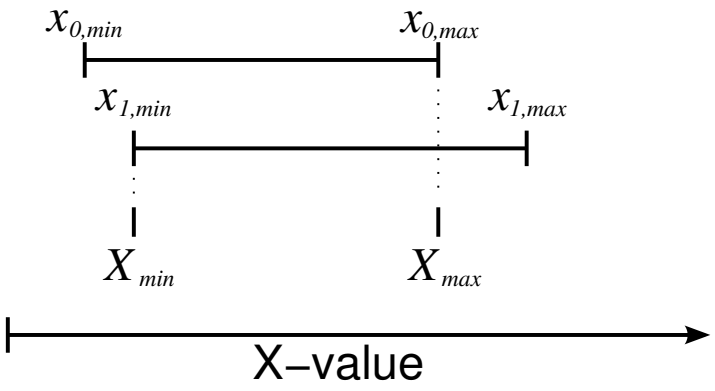


Figure 5.9: Filter region overlap

5.6 Efficient Local Adaptability

In order to adapt to the current environment and changes in the appearance of the road (for example, changes that occur when moving through shadows or from dirt onto gravel), the system samples the road in order to create a new colour filter. However, for much of the time, we can expect there to be only small changes in the appearance, essentially caused by the specific patch of road that is being sampled. To avoid the inefficient creation of new filters that do not actually help (because the colours they allow are largely already covered by existing filters), we introduce a means of comparing a new descriptor against those of the existing filters.

For two descriptors $\mathbf{D}_{R0}$ and $\mathbf{D}_{R1}$, and a tolerance $\Delta = [\Delta_x, \Delta_y, \Delta_i]$ we first define the following quantities:

$$\begin{aligned} x_{0,min} &= \bar{c}_0 - \Delta_x S_0 \\ x_{0,max} &= \bar{c}_0 + \Delta_x S_0 \\ x_{1,min} &= \bar{c}_1 - \Delta_x S_1 \\ x_{1,max} &= \bar{c}_1 + \Delta_x S_1 \\ X_{min} &= maximum(x_{0,min}\ ,\ x_{1,min}\ ) \\ X_{max} &= minimum(x_{0,max}\ ,\ x_{1,max}\ ) \end{aligned}$$

These are depicted graphically using number-lines in Figure 5.9 We then define the overlap of the two filters in the X channel as follows:

$$X_{0,overlap} = \frac{X_{max} - X_{min}}{2\Delta_x S_0}$$

with $Y_{0,overlap}$ and $I_{0,overlap}$ defined similarly for the remaining two colour channels. These measures describe the amount of overlap between two filters in each colour channel, normalised against the spread of the first filter. (It should be noted that this method of calculating overlap implicitly treats the filters as describing a cuboid, not ellipsoid, region of colour-space, and so is a simplified approximation to the true overlap.) A positive overlap indicates that there is a shared region in the relevant channel, while a zero or negative overlap indicates no common

coverage. We then define the similarity of the two filters, referenced against the first, as follows:

$$s_0 = minimum(\ X_{0,overlap},\ Y_{0,overlap},\ I_{0,overlap}\ ) \tag{5.7}$$

Two filters that share a common region in colour space (under the cuboid approximation) will give a positive similarity, while two that do not (even if there is overlap in one or two channels) will give a zero or negative similarity. A new filter is only created if the similarity thus defined is less than an empirically determined threshold (a threshold of 0.9 is used), meaning the region in colour space covered by the new filter is not effectively covered by an existing filter.

Filters are held in a list that is sorted based on the filter's utility u_F, defined as follows:

$$u_F = \frac{h_{total}}{1+t} \tag{5.8}$$

with h_{total} and t being the total hit count and age of the filter respectively, described in Sections 5.5.1 and 5.5.2. The utility is an indication of how useful, on average, a filter has been, with regards to its ability to consistently describe the road.

When a new filter is created (because the new statistics are different enough from all existing filters), the list of filters is first sorted by filter utility, the least useful filter is discarded and the new filter is added to the list. In this way, we provide adaptability to current conditions (through the creation of new filters) and a manageable temporal accumulation of knowledge (through a fixed-size set of filters that is modified as time passes).

5.7 Experimental Results

In this section, we present some experimental results that show the operation of the road-finding system. In the following figures, the left image shows the raw input image (taken from the centre camera) with the road-sampling region shown as a rectangle. The right image shows the resulting road likelihood map, with brightness indicating the likelihood of a point belonging to the road. The results shown were obtained using a tolerance of $\Delta = [3.0, 3.0, 4.0]$ (refer to Section 5.5.1).

In the first set of results, a total of three filters was used with a three-set k-means characterisation and forced filter creation. Subsequently, with the total filter count equal to the number of extracted descriptors (and all descriptors producing a new filter), the only filters used are those defined by the current characterisation.

In Figure 5.10 we see a fairly "clean" dirt road: the surface is predominately made of light-tan coloured dirt and the road is of fairly consistent width. The edges are still quite fuzzy, though, on the right foreground in particular. Dry leaf-litter is present on either side of the road and there is little greenery to be seen. In the distance, an obstacle (a large rock) sits on the road to the left. The likelihood map clearly shows the location of the road and notably (though not surprisingly) indicates that the (visibly different) rock does not belong to the road.

Figure 5.11 shows a slightly more varied road surface: both blue-grey gravel and light-tan soil are visible. The geometry is also slightly less consistent and, while the grass on the left provides a clear boundary, the dirt on the right leaves the definition of road edge quite

Figure 5.10: Single-frame example 1

Figure 5.11: Single-frame example 2

Figure 5.12: Single-frame example 3

Figure 5.13: Single-frame example 4

uncertain. In spite of both the varied surface appearance and the fuzzy border, the likelihood map shows that the road is well extracted, though it does tend to "bleed" slightly into the dirt along the right edge. If a robot were to traverse roughly down the centre of the extracted region, however, it would clearly be following a safe route.

The road shown in Figure 5.12 is quite different to previous results. There is a mixture of blue-grey gravel along the right and light-tan dirt intermingled with grass along the left. Neither edge is at all clear and the gradual blending into grass on the left makes that side particularly poorly defined. It is highly likely that, were ten people to label where the road boundaries were, ten different results would be obtained. Accordingly, the likelihood map is less clear than with previous results. In particular, there is a patchy region in the foreground to the right of centre where the road has failed to be extracted. Nevertheless, the overall road extraction is quite good considering the conditions.

Figure 5.13 shows another road segment with varied surface appearance, this time a mixture of light-tan and greyish-brown soils. Again, edges are unclear, especially on the left where a brown slight embankment makes the distinction between what is the road and what is not the road particularly unclear. In spite of this, the likelihood map shows that the road

Figure 5.14: Single-frame example 5

Figure 5.15: Single-frame example 6

has been successfully extracted with the exceptions of a few small regions on the edges.

The result shown in Figure 5.14 shows the system's ability to handle both bright regions and shadows simultaneously but also highlights a problem with such conditions. Because the sampling region contains both bright and shadowed areas, both of these very different road regions can be extracted successfully. While the bright and shadowed regions can be identified in the likelihood map, it can be clearly seen both are included in the overall extraction of the road and hence the road is well defined. The left edge and the right edge in the distance are also both well defined, though the right edge in the middle ground is not at all well defined and in fact in that region areas that are clearly not part of the road are incorrectly classified. This is the result of a combination of dark shadows falling on a region of ground that already exhibits only a small visual variation to the road surface. The shadow – or more correctly the shadow combined with very bright regions and the limited dynamic range of the camera – reduces the visual difference between road and non-road regions to such a low level that they become inseparable.

The result shown in Figure 5.15 is a good example of partial failure under adverse lighting conditions. While the left road boundary is accurately defined in the likelihood map, lens-flare

in the centre-right of the image causes that region to be poorly classified. Further problems are shown in the distance where a particularly bright region fails to be extracted.

In the next set of results, we show a short sequence of images taken as the robot progresses through an alternately bright and shadowed region. The three sets shown in Figures 5.16, 5.17 and 5.18 show the effects of using a history of filters and of using similarity-based filter creation. In these sets, processing begins with the first image, that is, there exist no previous filters and hence no history before the first image.

In Figure 5.16 a total of three filters is used along with a three-set k-means routine and filter creation is forced. This means that, firstly, the k-means road characterisation uses three clusters and so three road descriptors (see Section 5.4.2) are produced. Secondly, because filter creation is forced and there are a total of three filters in operation, the filters used to extract the road are determined solely by the current characterisation; there is no history.

In Figure 5.16(a) the sampling window lies almost solely in a shadowed region and so other similar shadowed areas are shown as belonging to the road, while the bright regions that appear quite different are not. In Figures 5.16(b) and 5.16(c) enough of the bright region is contained in the sampling window to cause both shadowed and bright regions to be correctly classified as belonging to the road. In Figures 5.16(d), 5.16(e) and 5.16(f), however, only the bright region falls within the sampling window and, without history, the appearance of the shadowed region is "forgotten", leading to a sparse and poor extraction of the true road.

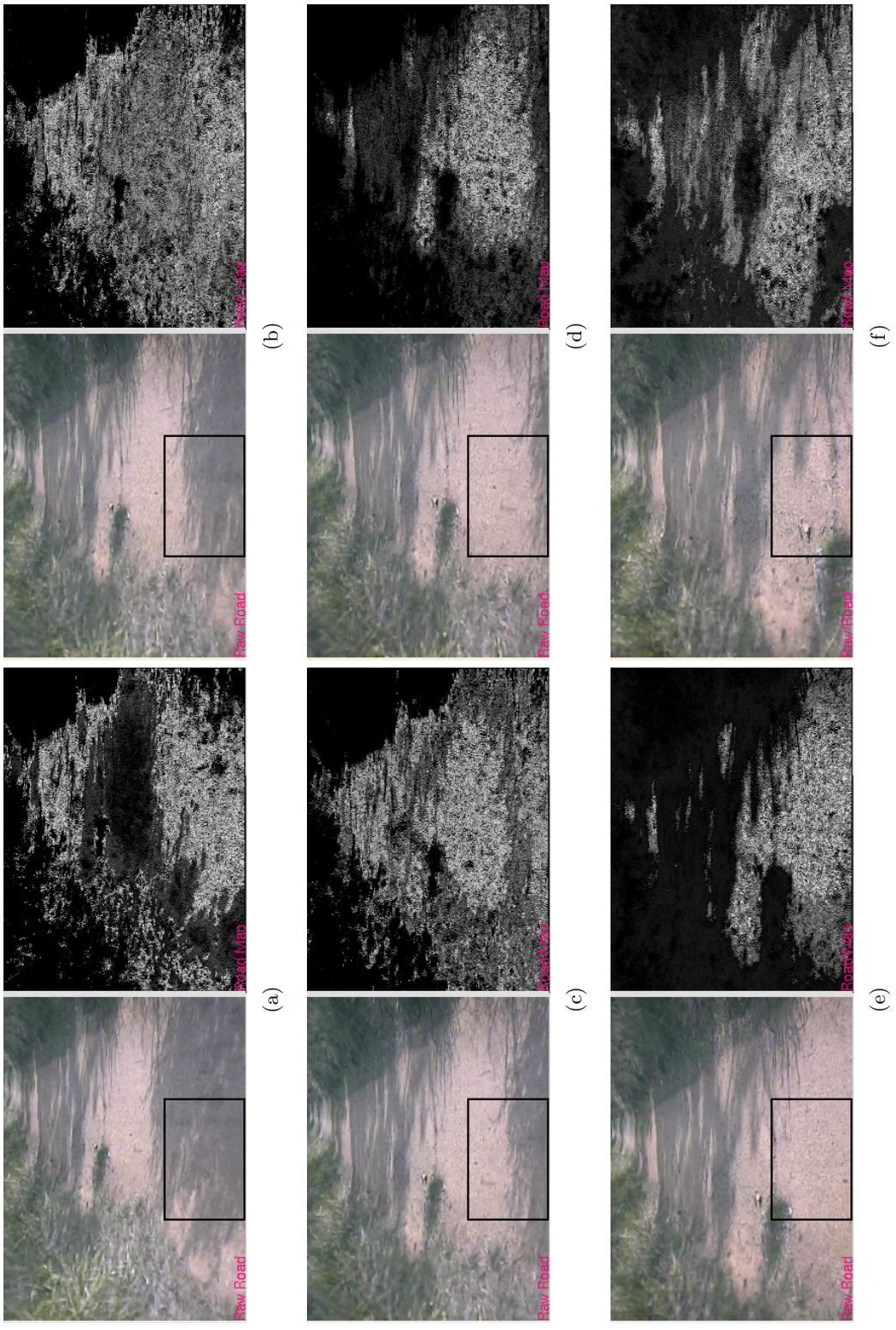


Figure 5.16: Three filters, three-set k-means, forced creation

In Figure 5.17 a total of six filters is used along with a three-set k-means routine and filter creation is forced. Because the total number of filters (six) is greater than the number of k-means extracted road descriptors (three), there is a filter history. Again, though, filter creation is forced, so the filters in operation are the three newly created filters plus three previous filters.

As previously, in Figure 5.17(a) there exists no history (as this is the first image), and so only the sampled shadowed region is correctly classified. In Figures 5.17(b) and 5.17(c), as before, bright and shadowed regions are both being sampled adequately and accordingly both bright and shadowed regions are correctly classified. The effect of maintaining a history becomes apparent in Figure 5.17(d) where, in spite of the shadowed region no longer being sampled adequately, the previous set of filters still allow the shadowed regions to be correctly classified, leading to a good road extraction and a clear improvement on the previous result. In Figures 5.17(e) and 5.17(f), however, we see the limitations of creating new filters without regard for their ability to provide new information. The appearance of the ground within the sampling windows of Figure 5.17(e) is very similar to that in Figure 5.17(d), but without taking this similarity into account, new filters are created that cover a very similar region in colour space and hence provide very little additional benefit. With a fixed number of filters in operation, previous filters that matched the shadowed regions are discarded and so the ability to recognise previously seen shadowed regions as part of the road is lost. The same situation occurs in Figure 5.17(f). In these circumstances we are no better off than when we had no history.

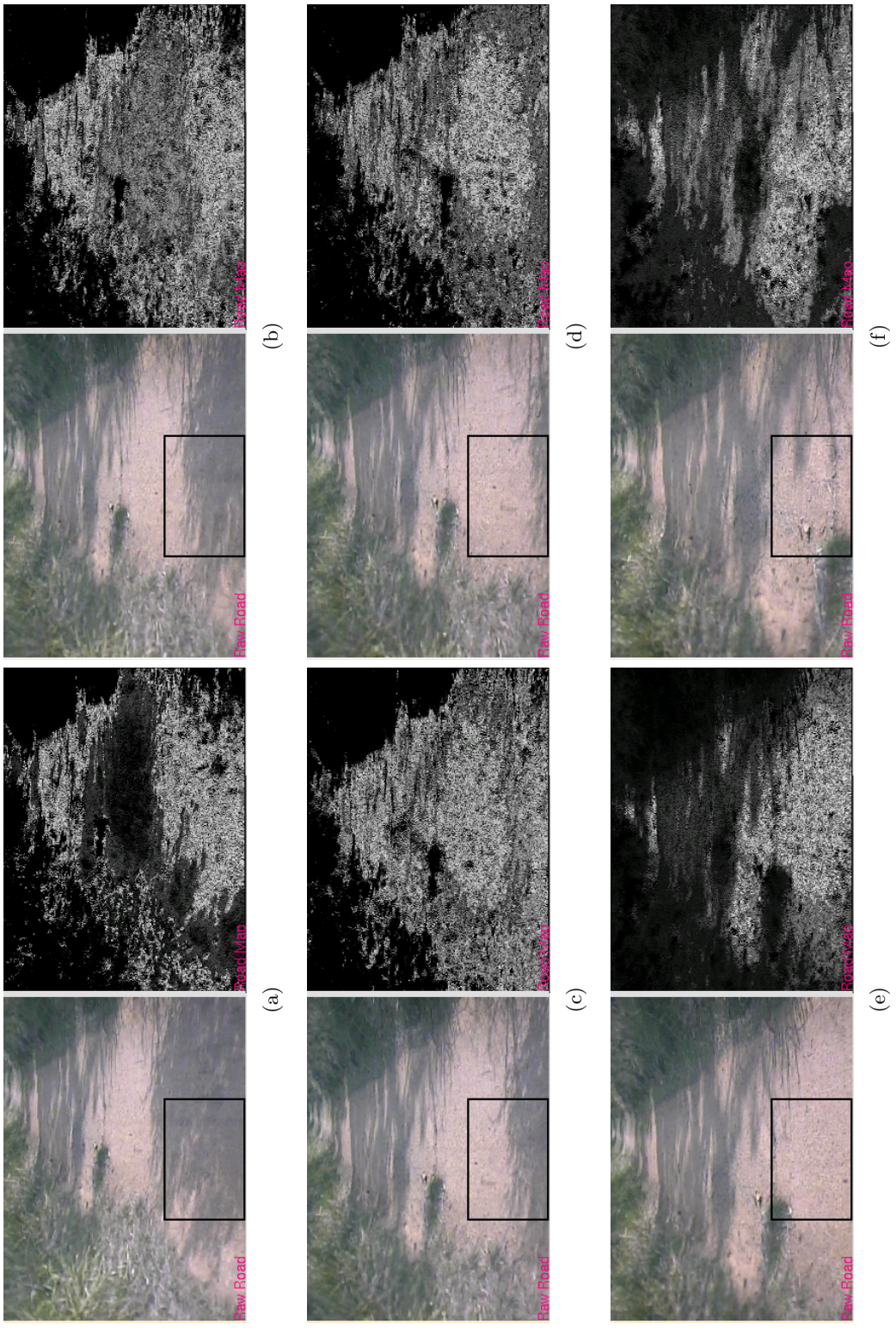


Figure 5.17: Six filters, three-set k-means, forced creation

Finally, in Figure 5.18, a total of six filters is used with a three-set k-means characterisation, but this time filter creation is based on similarity: only if a new road descriptor is dissimilar enough from existing filters is a new filter created. A similarity threshold of $s_0 = 0.9$ is used (see Section 5.6). We therefore have an adaptive history of filters.

Once again, there is no history available in Figure 5.18(a) and so only the sampled shadowed region is correctly classified. As previously, both bright and shadowed regions are sampled in Figures 5.18(b) and 5.18(c), thereby allowing good extraction of these visually distinct road regions. In Figure 5.18(d) the shadowed region is no longer effectively sampled, but the filter history allows the previously seen shadowed road to be correctly classified, as was the case in the previous set of results. The great improvement in performance achieved by selective filter creation is shown in Figures 5.18(e) and 5.18(f). The shadowed region is no longer being sampled and was last "seen" two frames ago in Figure 5.18(e) and three frames ago in Figure 5.18(f). However, because new filters are only created (and hence old filters discarded) if new descriptors are significantly different from those that define the existing filters, and there is little difference in the sampled road's appearance in Figures 5.18(d), 5.18(e) and 5.18(f), the filters corresponding to the shadowed region remain in history, allowing full and accurate extraction of the entire road.

A collection of additional experimental results are given on the accompanying CD-ROM, referred to in Appendix .6.

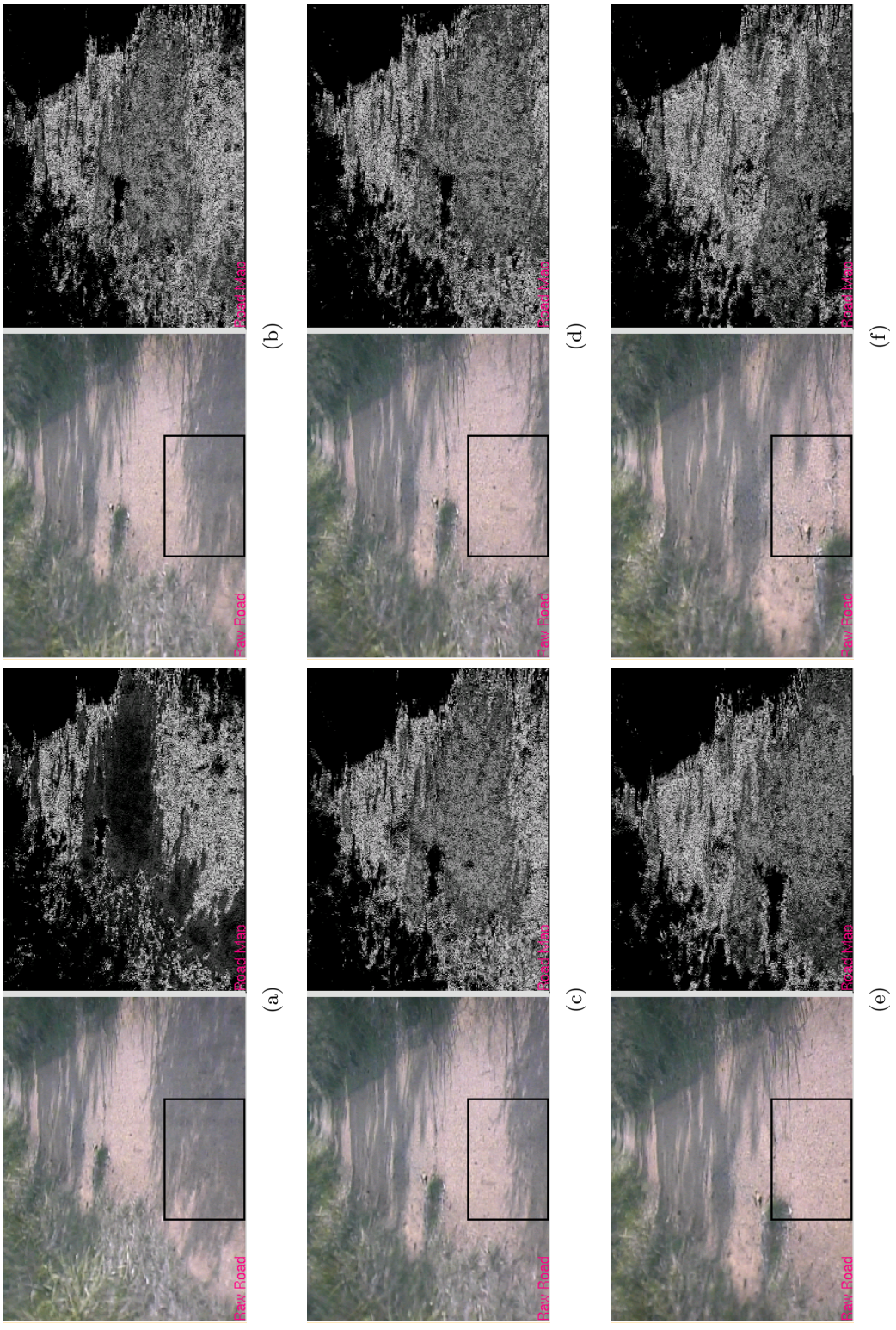


Figure 5.18: Six filters, three-set k-means, similarity-based creation

5.8 Conclusion

This chapter presented in detail one of the core problems this thesis is attempting to solve, that of finding poorly structured dirt roads in bush environments. The difficulties in doing so were described and notably include the lack of consistent geometry, markings or predictable surface features that are typically exploited by urban road-finding systems. The solution presented was a system that is intended to firstly characterise the appearance of a degraded dirt road and then extract the road from the surrounds in an image, creating a likelihood map that describes the likelihood of a visible location belonging to the road. The characterisation produces a set of statistical descriptions of a sampled region of the image that is assumed to contain the road, in terms of pixel colour. In order to provide robustness against adverse lighting (for example, the differences between dark shadowed areas and well-light regions), the colour description used is one that effectively separates intensity from colour, allowing intensity to be treated with wider tolerance and hence allow greater variation. To accommodate roads with significantly different surface materials (for example, brown mud and blue-grey gravel), the k-means algorithm is used to produce a set of statistical descriptors instead of using just one descriptor. Road extraction is done with a colour-filtering procedure, where each of a set of filters represents an ellipsoid in the chosen colour-space, defined by a previously estimated descriptor. Filters are ranked according to their utility, and when a new filter is created, the least useful existing filter is discarded, keeping the number of filters (and associated processing costs) manageable. Additionally, new filters are only created when the defining statistics are significantly different from all existing filters, thereby reducing redundant information and maintaining diversity. Finally, a filtering method is used reduces computation by selectively sub-sampling the image and only processing at finer resolution when a region of interest is discovered. The experimental results presented show the system's ability to handle roads with mixed (and visibly different) surface materials comfortably and show how it is able to handle both strong lighting and strong shadows simultaneously, though is not immune to extreme lighting conditions. Also shown is the significant improvement provided firstly by the use of a history of filters and further by the selective creation of new filters based on their ability to provide new information.

Chapter 6

Local Path-Planning using Likelihood Maps

> “ *For every path you choose, there is another you must abandon, usually forever.* ”
>
> **Joan D. Vinge, 1948–present**

Figure 6.1: Where do we go now?

It would be difficult to call a mobile robot intelligent if it was unable to decide, at some level, where to go next. In a small-scale, localised sense, this means it needs a representation or map of its environment (either provided or determined *in situ*), a notion of where it is (possibly embedded in a robot-centric map) and the ability to plan a path from its current position to some goal (again, either provided or self-determined). In the context of this research, the robot's intermediate goal is to find its way along a dirt road, avoiding obstacles as it does so. Previous chapters examined firstly how obstacles may be detected (Chapter

4) and secondly how a poorly-structured, unsealed and unmarked dirt road can be found (Chapter 5). These features are given in the form of likelihood maps, one depicting the likelihood that each point is part of an impassable obstacle, the other the likelihood that each point is part of a navigable road surface. Together these provide the information necessary to construct a robot-centric map. What remains, then, is the generation of a local path that will allow for safe, incremental traversal of the road and the means by which that path is followed.

6.1 Introduction

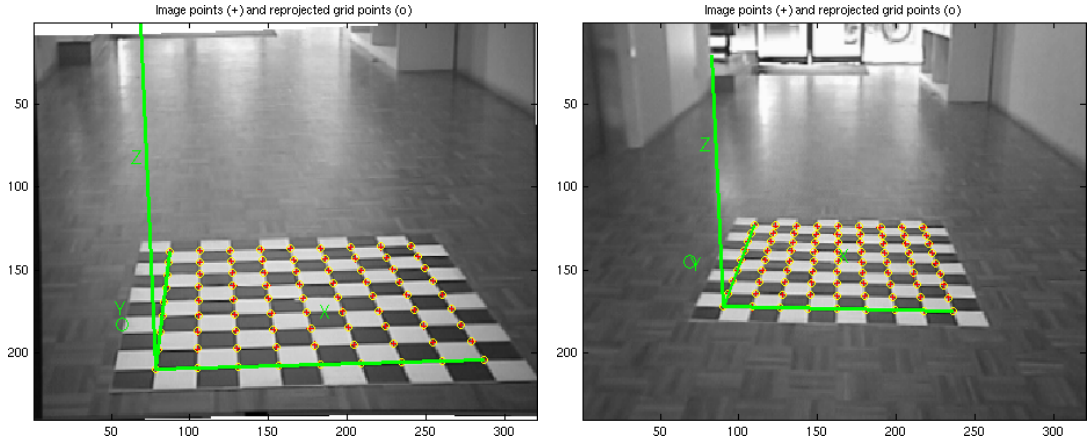
The material presented in Chapter 4 and Chapter 5 describes the means by which two forms of local environmental maps may be built. One describes the likelihood of a location containing an impediment to motion (that is, an obstacle), representing areas we wish to avoid. The other describes the likelihood of a location being part of the road, representing areas that are favourable.

Under most circumstances, we would expect these two feature-spaces to be complementary: in wanting to traverse along the road, non-road areas are to be avoided and obstacles on the road are clearly not part of the road itself. Accordingly, it could be assumed that one map could be used to generate the other and hence we have redundant information. However, this is not necessarily the case. For example, in certain situations we may need to deviate off the road, either temporarily to avoid a large obstacle on the road or for longer periods to undertake cross-country travel if higher-level planning decides that is the best choice. In either case, safe terrain is *not* strictly road terrain, it is simply ground that contains no obstacle and hence is physically traversable. On a more subtle level, some obstacles *are* part of the road, such as potholes and similar abrupt changes to the ground level. Impediments such as these would likely be included as road regions on the road likelihood map while being labelled as obstacles on the obstacle likelihood map. Here there is a clear difference between the two maps, and the desired behaviour is to give preference to the obstacle map.

These two compatible but effectively independent feature spaces form the basis of our navigation workspace and are the data with which we will determine the robot's desired trajectory. They are used to satisfy the underlying goal of driving along a road while avoiding obstacles.

To make use of this data, two basic steps are required: the fusion of data from multiple sensors and the generation, evaluation and selection of paths and corresponding immediate goals. Within the current system, only two data sets are present (representing obstacles and the road), but the framework used implicitly handles any number of sets, meaning, with the current setup, data can be ignored if required (for example, if intending to travel off the road, road information can and should be ignored), or, if sensor modalities or constraints are extended in the future, additional data can be incorporated.

The first step, data fusion, uses information gained through system calibration to allow sensor information taken from different view-points to be combined. This is an important step and ensures that the final data-set, the overall map, accurately represents the environment. Without this step, data from different sensors would not be spatially aligned, leading to poor or even dangerous path selection. The underlying goal is the creation of what is essentially a two-dimensional grid-based map that is a variation of a simple occupancy grid. Data taken from different sensor systems (in this case stereo-based obstacle detection and colour-image-



(a) Left image (b) Centre image

Figure 6.2: Once scene, two perspectives

based road finding) is firstly mapped into a common frame of reference (an overhead, birds-eye representation) and then combined to give a single map.

The second step aims to both determine an intermediate goal location and fit a path model to the resultant map, based upon some simplifying assumptions. The typical high-level aim of the robot is to drive along the road, but to do this it must determine a series of immediate goals that will serve as waypoints as well as trajectories that will guide the robot to the waypoint. Waypoints are local goal locations that effectively answer the recurring question of "*where do I go next?*". The central assumption being made is that the road generally flows away from the robot in approximately the direction it is facing. Accordingly, the chosen path is one that flows away from the robot in the same direction, keeping to areas that belong to the road and are obstacle-free. If the mission dictates, through high-level planning, that cross-country traversal is required, the road-likelihood information is ignored, but the remaining constraints apply. It is still assumed that the desired direction of travel is approximately the direction the robot is facing. This can easily be met by having the robot point towards its goal before proceeding cross-country and reorienting intermittently if necessary. The assumptions of flow and direction of travel are necessary to constrain the possible path and waypoint choices.

In order to evaluate potential paths, path length, smoothness and overall amenability to traversal (in terms of obstacles and the road) are used as criteria, with a long, smooth path with high overall amenability considered best. A long path means a greater distance between waypoints and therefore more efficient planning (as each waypoint is only determined after a new scan). A smooth path requires small correctional changes to motion and reduces the likelihood of producing large errors in motion. Finally, a path that is amenable with regards to the location of the road and obstacles is a safe path and increases the likelihood of unhindered traversal.

6.2 Image Registration

The path that we wish the robot to take is determined by two essentially independent factors: the physical traversability of the terrain (as determined by the presence of obstacles, detected

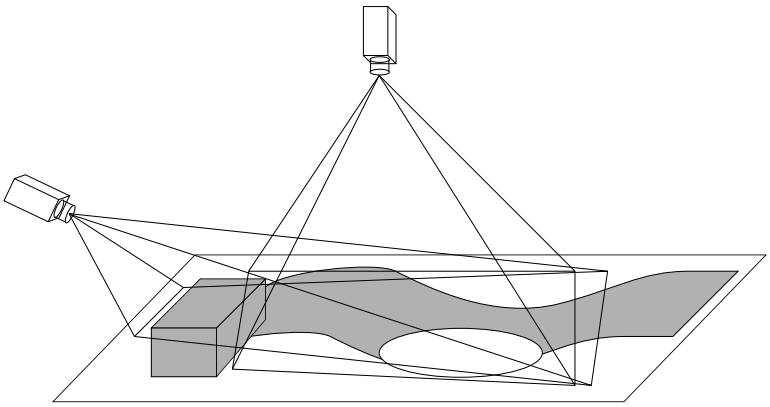


Figure 6.3: One scene, two viewpoints

by the stereo system, Chapter 4) and the location and flow of the road (detected by the dirt-road finding system, Chapter 5). These features are given in the form of likelihood maps that show the likelihood of the presence of either an obstacle or the road at each point in an image. However, to allow for a more general configuration (particularly the use of a relatively wide-angle lens for road-finding and comparatively narrow-angle lens with less distortion for stereo), the likelihood maps originate from different perspectives (that is, different cameras). Figure 6.2 shows a calibration scene containing the calibration grid as viewed from the left stereo camera (the reference frame for stereo processing) and also from the centre camera (used for road-finding). The difference in images (largely due to the different focal-lengths being used) is clear. To combine the maps, we first need to represent the information in a single, consistent coordinate system; that is, the maps need to be registered through re-projection.

6.2.1 The Geometry of Image Re-Projection

Figure 6.3 shows diagrammatically a scene visible to two cameras. In this section we present the mathematics that can be used to give the location of a point in one image as it appears in the other. In this way, information gained from one image can be expressed in the reference frame of another image, allowed data from different sources (that is, different cameras) to be meaningfully combined.

In the setup being used, a point visible to a camera can be represented in a number of reference frames. The first is the camera's unique reference frame, where the origin is the camera's centre of projection, the positive z-axis is aligned with the optical axis, and the X-Y plane is parallel with the image plane. The second is a local ground reference frame, whose X-Y plane is parallel to the calibrated flat ground surface and whose z-axis extends upwards. This reference frame is obtained via calibration (as shown in Figure 6.2) and may be shared by multiple cameras. (The two stereo cameras and the road-finding camera share a common ground-reference frame, but the two visual odometry cameras have their own, independent ground-references). Finally, there is the robot frame of reference. This reference frame is common to all cameras.

The relationship between a point in the camera reference frame $\boldsymbol{p}_c$ and a point in the

ground reference frame $\boldsymbol{p}_g$ is given by

$$\boldsymbol{p}_c = \mathsf{R}_{cg}(\boldsymbol{p}_g + \boldsymbol{T}_{cor}) + \boldsymbol{T}_{cg} \tag{6.1}$$

$$\implies \boldsymbol{p}_g = \mathsf{R}_{cg}^{-1}(\boldsymbol{p}_c - \boldsymbol{T}_{cg}) - \boldsymbol{T}_{cor} \tag{6.2}$$

where R_{cg} and $\boldsymbol{T}_{cg}$ are a rotation matrix and translation vector, respectively, describing the orientation and position of the calibrated ground reference frame with respect to the camera and $\boldsymbol{T}_{cor}$ is a translation to correct for a difference between the calibrated ground origin and the true ground origin. This difference is simply a side effect of the calibration process whereby, as shown Figure 6.2, the calibrated origin is displaced from the actual grid origin. The parameters R_{cg} and $\boldsymbol{T}_{cg}$ are obtained via the "*Camera Calibration Toolbox for Matlab*" [10] while $\boldsymbol{T}_{cor}$ can simply be measured from the grid itself.

During calibration, all ground-based calibration grids are aligned such that there exists only a translational difference (that is, no rotational difference) between them and the robot origin, defined to be the point on the ground underneath the front, driven wheels located centrally, as shown in Figure 6.4. Therefore, the transformation from local ground reference to global (robo-centric) robot reference-frame is given simply as follows:

$$\boldsymbol{p}_g = \boldsymbol{p}_R + \boldsymbol{T}_{gR} \tag{6.3}$$

$$\implies \boldsymbol{p}_R = \boldsymbol{p}_g - \boldsymbol{T}_{gR} \tag{6.4}$$

where $\boldsymbol{T}_{gR}$ is the translation between the ground (grid) origin and the robot origin.

Using these relationships and the calibrated parameters, a known point from any one camera can be reprojected into any other camera's frame of reference, thereby registering images.

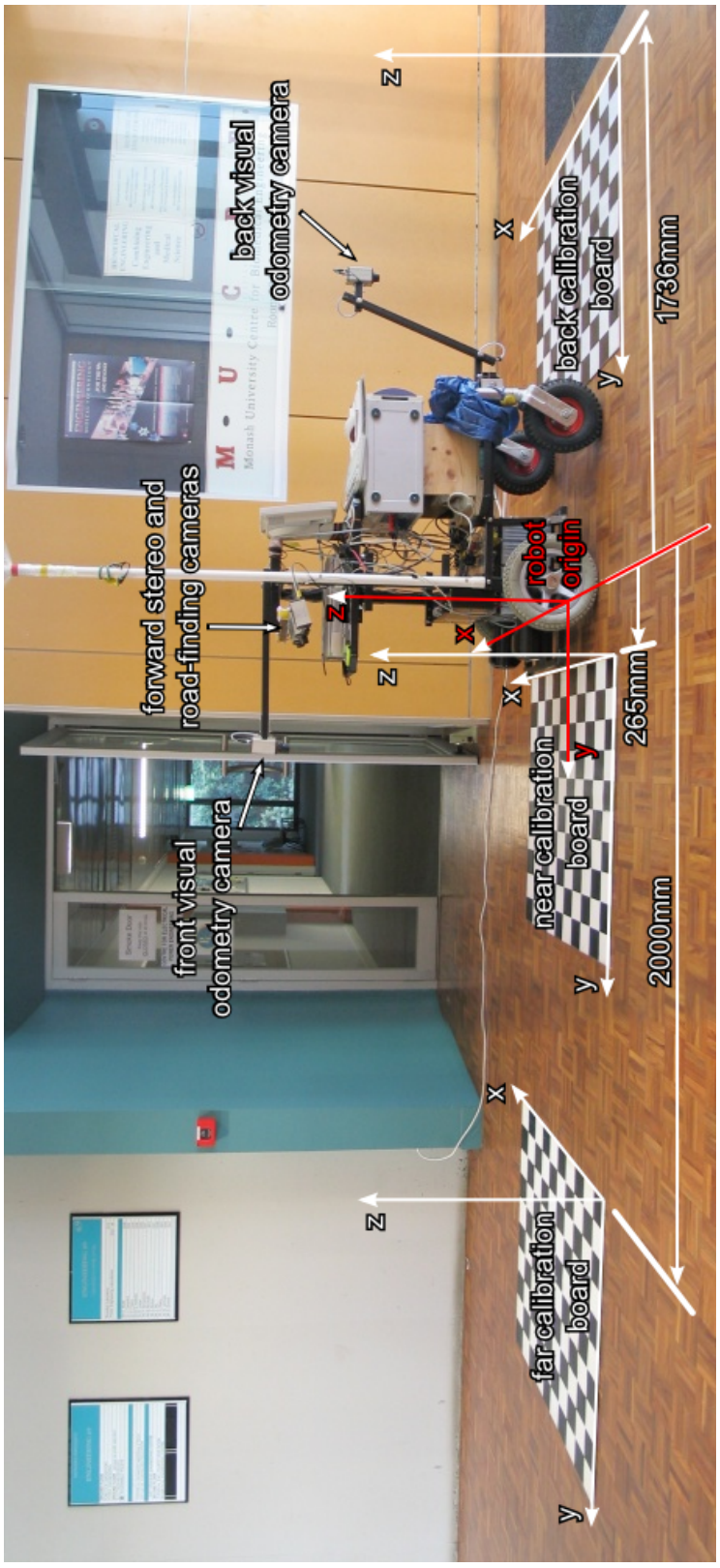


Figure 6.4: The ground calibration setup

For the purpose of local path-planning, we desire a robo-centric frame of reference that clearly represents the space around the robot within the sensing field of view. This perspective is represented by a virtual camera that is oriented to give a birds-eye view of the environment, focusing on the region directly in front of the robot.

A point in the virtual camera $\boldsymbol{p}_v$, is similarly related to the robot coordinate system by R_v and $\boldsymbol{T}_v$, and can be described by

$$\boldsymbol{p}_v = \mathsf{R}_v \boldsymbol{p}_R + \boldsymbol{T}_v$$

We can then re-project a point in a camera's reference frame into the virtual reference frame via

$$\begin{aligned} \boldsymbol{p}_v &= \mathsf{R}_v \boldsymbol{p}_R + \boldsymbol{T}_v \\ &= \mathsf{R}_v(\mathsf{R}_{cg}^{-1}(\boldsymbol{p}_c - \boldsymbol{T}_{cg}) - \boldsymbol{T}_{cor} - \boldsymbol{T}_{gR}) + \boldsymbol{T}_v \end{aligned} \tag{6.5}$$

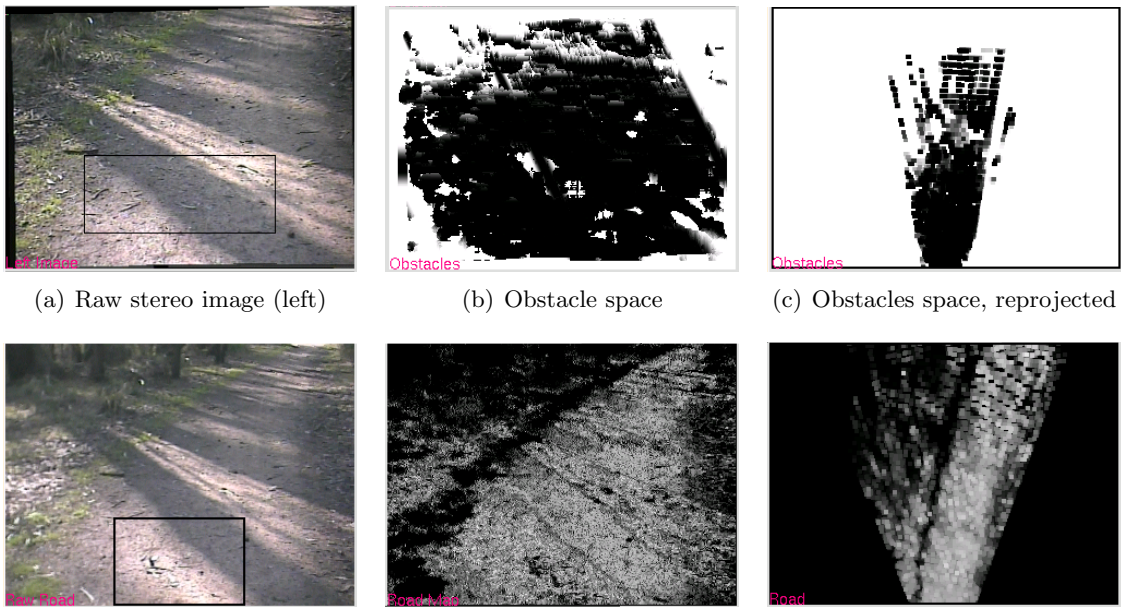
To use Eq^n 6.5, we need the position of each point expressed in the camera's reference frame. For the stereo system, we have this information: stereo processing provides the third and otherwise unknown depth coordinate. For the road-finding system, this information is not directly available. We therefore make the assumption that points seen in the road-finding camera lie on the calibrated ground plane. However, with the road-finding camera's coordinate system known relative to the stereo camera's reference frame, an intermediate re-projection can be performed that uses the known stereo-derived depth information to augment the assumed depth information where possible.

To achieve this, we first back-project stereo points into the robot reference frame via Eq^n 6.2, then forward-project these points into the road-finding camera's coordinate system via Eq^n 6.1. This then gives us the required depth information for all points both visible in both cameras and whose stereo-based depth can be determined, these being the points we most importantly need to register.

Figure 6.5 depicts the re-projection of an obstacle map and a road map onto a virtual camera that is looking down on the scene in front of the robot. In these (and the following) figures of processed maps, light areas indicate areas of high likelihood, while dark areas represent low likelihoods. Raw input images are shown in the left column (Figure 6.5(a) and Figure 6.5(d)), processed maps in the middle (Figure 6.5(b) and Figure 6.5(e)), and reprojected processed maps in the right column (Figure 6.5(c) and Figure 6.5(f)). Because of the change in perspective, points are less densely spaced at the top of the reprojected maps (representing the area most distant from the robot), and so morphological operations are performed (the obstacle map is eroded while the road map is dilated) as a post-processing operation to compensate.

6.3 Amenably Traversable Terrain

Our goal is to drive the robot over what is considered to be safe and efficiently traversable terrain, that is, along the road while avoiding obstacles. Having expressed both obstacle and road likelihood maps in a common and consistent frame of reference, we now look to combining these likelihoods to determine an overall map describing terrain that is amenable to safe and efficient traversal. The likelihood of a location having feature f (where f is either obstacle or road), is denoted as L_f and is in the normalised range $[0, 1]$. However, we



(a) Raw stereo image (left) (b) Obstacle space (c) Obstacles space, reprojected

(d) Raw road image (e) Road space (f) Road space, reprojected

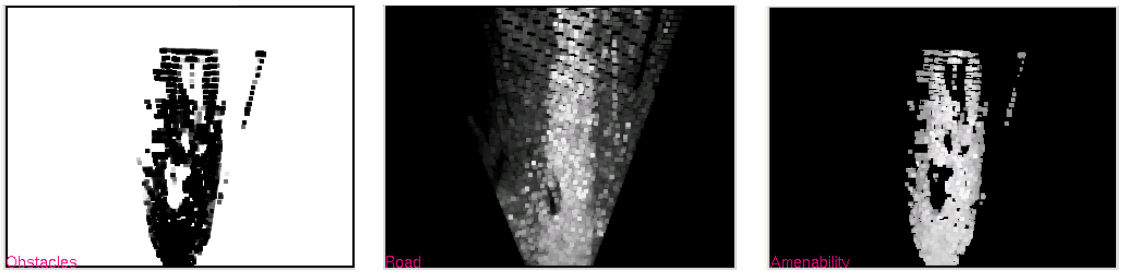
Figure 6.5: Re-projection of obstacle and road spaces onto virtual camera

wish to provide a means by which the relative importance of each feature can be adjusted. Typically, we wish to place a slightly higher importance on avoiding obstacles, as traversal into an obstacle is likely to present a greater impediment to motion (and present a greater risk of mission failure) than simply straying off the road temporarily. However, under certain circumstances (for example, if we are required to travel cross-country, that is, off the road), we would wish to ignore the road information. Each feature is therefore assigned a simple weight, denoted γ_f, and the overall amenability to traversal, A_{trav}, is given by

$$A_{trav} = \gamma_{obs} L_{obs} + \gamma_{road} L_{road} \tag{6.6}$$

Accordingly, γ_{obs} will be negative and γ_{road} will be positive. An amenability below zero then indicates terrain we should strongly avoid, zero indicates somewhat neutrality with regards to traversal, and positive indicates the terrain we most strongly wish to traverse. Applying Eq^n 6.6 to each point in the re-projected likelihood maps produces an overall amenability map, which we can then use to deduce a safe path. Figure 6.6 shows an obstacle and a road likelihood map, and the resultant amenability map.

The amenability map is in essence a finely-grained occupancy grid where the value of each cell is the likelihood of that cell being safely traversable. This is considered to be both a feasible and effective representation because the map is only intended to be kept for a short time and used locally, and so problems associated with scaling (and the corresponding increases in storage and processing requirements) are not likely to occur.



(a) Obstacle-space (b) Road-space (c) Amenability

Figure 6.6: Obstacle and road likelihood maps and the resultant amenability map

6.4 Finding Potential Paths

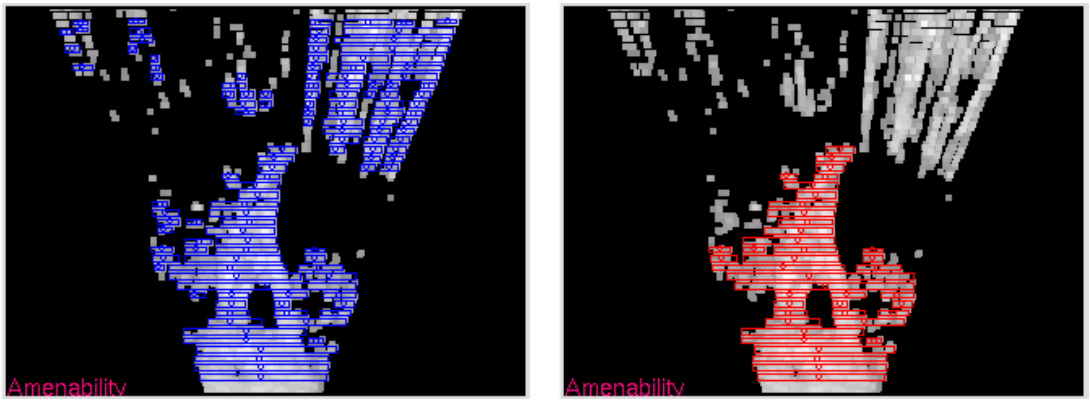
Having determined for each point in the map an amenability to traversal, we are now in a position to find potential paths, compare them, and determine the best candidate, which then determines where the robot's next waypoint will lie. This process consists of three steps: finding contiguous regions in the map, linking them to form potential candidates, and fitting a model to each candidate to determine its suitability.

Traditionally, maps based on occupancy-grids are typically coupled with planners based on the distance-transform or potential-field concepts, where the path from known starting location (that is, the current position) to known destination is incrementally built, for example, by performing a steepest descent operation using the distance transform or by iteratively determining the resultant "force" being applied to the robot by the attraction of a goal and the repulsion of obstacles in a potential field. In either case, the destination must be known (whether provided or determined) before a path can be planned. In the context of following dirt roads or attempting straight-line cross-country traversal, these intermediate goals must be determined from an examination of the locally produced amenability map, applying the high-level goal of either following the road or maintaining a bearing, to find a destination that lies within safe (that is, amenable) space and is in the appropriate direction (either the direction of the road or the desired bearing), but is located some distance away. The tasks of determining an intermediate destination and finding a path to that destination are therefore tightly coupled. Accordingly, a different approach to finding a path is adopted where no specific destination is used. Instead some simple geometric and environmental assumptions are made which allow the generation of multiple paths which can then be compared to find what is considered the best option.

6.4.1 Sliced Segments

With our underlying goal being to drive along a road, the assumption is made that the road we wish to follow, and hence the path we will wish to take, flows generally away from the robot in approximately the direction it is facing. This assumption, which will typically be valid whenever the robot is on the road and either already travelling along it or at least facing the direction it wishes to travel, allows the process of determining the path and intermediate goal to be simplified.

Following this assumption, the map of amenable terrain is separated into slices: horizontal



(a) All segments (b) Merged and culled segments

Figure 6.7: Segments: before and after merging and culling

regions that lie perpendicular to the direction the robot is facing. We then segment the map into contiguous regions within each slice, looking for what should be safe regions of terrain that may be separated by potentially unsafe areas. Segments are described by their centre-of-mass, total mass and area, where member pixels contribute according to their amenability (that is, the total mass is the sum of all contributing pixels' amenabilities, and the centre-of-mass is the amenability-weighted mean position of contributing pixels). Only segments above a minimum size (in both area and mass) are considered further. Within a slice, segments that are separated by less than a maximum distance are merged to allow for small discontinuities that should not impede motion. For example, a small tuft of green grass in the middle of a brown dirt road would not impede motion, but would likely be considered to not belong to the road, given the difference in appearance. Additionally, starting closest to the robot, the adjacency of segments between slices is examined. A gap between segments is not allowed, and any segment that is not directly adjacent to another segment that is closer to the robot is culled.

Figure 6.7 shows an example of a segmented amenability map. In Figure 6.7(a), all segments are shown, each described by a bounding rectangle and a circle at its centre of mass. Figure 6.7(b) shows the merged and culled segments, described similarly. As can be seen, many segments in the upper part of the image that are clearly not part of the road have been removed due to their separation from the main road body. Along the main road body, a number of small segments can also seen to have been merged with horizontally adjacent segments when the separation is very small.

6.4.2 The Graph of Good Ground

To maintain a safe path, we wish, as much as possible, to stay on or in the regions we know to be safe and amenable to motion. Other than the assumption that the road flows generally away from the robot, we do not know where it lies. It may veer to the left, it may veer to the right, it might fork or it may be interrupted by obstacles. To accommodate this, we create a graph from the derived segments. The graph is rooted at the robot, and extends away, linking increasingly distant segments that touch but are within adjacent slices. In the case of

Figure 6.8: The path graph

a fork, for example, we will have a single segment in the slice at the base of the fork but two, horizontally separated segments in the next slice, where the road has forked. The segment at the base will then be linked separately to the segments in both arms of the fork, and so on. Similarly, an obstacle in the middle of the road will produce a graph that forks around the obstacle and recombines beyond it. The centre-of-mass of each included segment forms a node in the graph, such that the graph flows between points of maximum amenability (following the assumption that the centre-of-mass is the most amenable point within a segment). By construction, the graph is directed and acyclic, as each new node that is added may be the child of multiple existing nodes, but is the parent of none.

This graph represents the set of potential short-term paths that conform to our requirement of staying on safe terrain. The adjacency requirement means a robot moving along the graph should never travel over unsafe terrain.

Figure 6.8 is an example path graph, showing path segments (as previously) as the resultant graph that connects the segment centres-of-mass.

6.4.3 From a Graph to a Trunk: Evaluating Potential Paths

The graph that we have created describes, in general, a whole series of potential paths, any of which may be taken. We must therefore reduce this graph to a single trunk that represents the desired path. To do this, we need a way of comparing potential paths against each-other, according to some metric, allowing them to be ranked and a most favourable candidate to be chosen.

As much as possible, for both simplicity and efficiency, we would like the robot to take a smooth path that takes the robot as far along the road as possible. By smooth we mean a path that will require only small steering changes, reducing the chance of errors associated with large or rapid changes in the direction of the robot. The desire for a relatively long path trades off the ability to react to environmental changes against the efficiency of needing many

(computationally) costly scanning and planning operations. In the bush, it is assumed that the physical environment is largely static. While wind can cause rapid motion of branches and shifting clouds can quickly change lighting and shadows, for example, these changes do not have a significant effect – if any effect at all – on where it is safe to travel. Moving obstacles are also unlikely to be encountered, largely because human and vehicular traffic is very low in these areas. Those obstacles that are likely to be encountered – animals, such as kangaroos – typically move faster than the robot could realistically react to, and so an attempt to accommodate them would lead to ineffective planning.

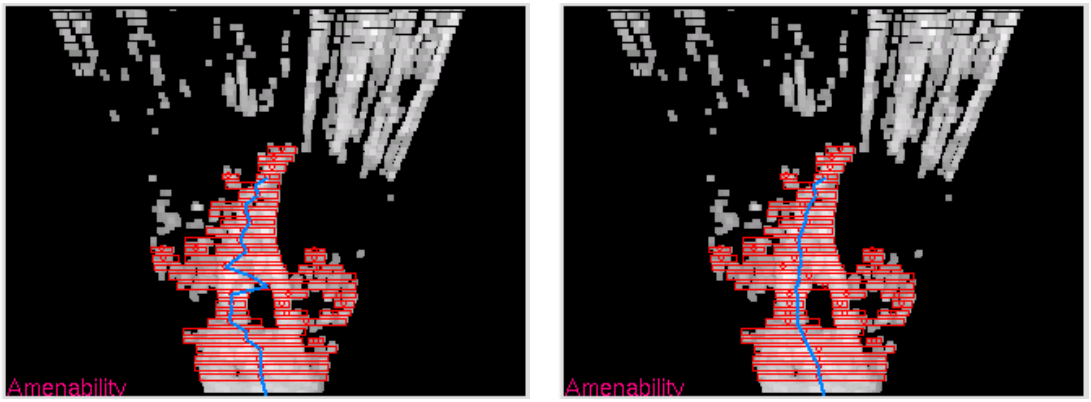
We would also like the robot to travel where it is considered safest, meaning traversing a path that is highly amenable (according to obstacle and road detection) and that provides physical safety by being wide and offering large buffers on either side of the robot.

To find a potential path, we begin at the root of the graph and work outwards, performing a breadth-first traversal. At each stage, the current node is appended to a list of nodes that describes the path taken to reach the current node. It is this list of nodes (where each node is a segment in the amenability map) that is used to fit a path model for evaluation. A limit is placed on the total number of paths allowed to be evaluated, in order to place an approximate bound on the processing time required when the amenability map is poorly constructed and produces a dense, highly-connected graph. By using a breadth-first traversal, we favour short-distance, wide-area exploration of paths instead of long-distance, narrow-area exploration, as would be obtained with a limited depth-first traversal.

To create a smooth path, a small-window smoothing operation is first performed on the centres-of-mass of the contributing segments. This is primarily done to remove large jumps between adjacent segment centres-of-mass that typically occur when a narrow segment appears adjacent to a wide one. For example, at the point of splitting, a fork in the road will produce two segments that will each be approximately half the width of the preceding segment. This is depicted in Figure 6.9, where segment centres-of-mass are shown as circles and the trace of contributing segments is shown as a solid line. In Figure 6.9(a) the trace undergoes a sharp change in direction at the point of forking, while in Figure 6.9(b), with smoothing, the change in direction is much less extreme. In this usage, smoothing serves to avoid potential problems associated with rapid changes in robot direction and the potentially dangerous situation of cutting corners.

We choose to model a potential path as a simple second-order polynomial, reflecting the desire for a path that can accommodate road curvature yet still produces a smooth trajectory. In fitting the polynomial to a potential path we use the error – the difference between the model and the contributing nodes' locations – as an indication of how smooth that path is (compared to the idealised polynomial path).

There is one other factor that needs to be considered when selecting a final path: direction. In some situations, most notably when facing a fork in the road, a decision needs to be made (typically by a higher-level, global path-planner) as to which fork to take. At other times, the robot may be traversing cross-country, and needs to maintain a given bearing, again provided by a higher-level planner. In the more general case, we will simply wish to maintain a consistent trajectory and provide some immunity to instabilities and vacillation that may occur when the choice of path is not distinct. To allow for these types of control, we introduce the notion of path bearing. In evaluating a path, in addition to the second-order polynomial, a first-order polynomial is fitted, and the gradient of this model is used to determine the overall bearing of the path, relative to the robot. We then have a form of control with the bearing error, equal to the absolute difference between the path bearing and the desired bearing. The



(a) Un-smoothed (b) Smoothed

Figure 6.9: Original and smoothed potential path

desired bearing is either given by a global-planner or simply equal to the previous bearing in order to produce a temporally consistent choice of path.

For a given potential path, we have a number of defining characteristics. The total amenability A is the sum of all contributing segment masses. The displacement d is the straight-line distance between the first and last contributing segments. The error ϵ is the difference between (smoothed) segment centres-of-mass and the fitted model. The minimum buffer b is the minimum distance between the fitted model and horizontal segment boundaries. The average width $\bar{w}$ is the average width of contributing segments. The bearing error θ is the absolute difference between the approximate path bearing and the desired bearing.

The overall fitness of this path is defined by:

$$f = c_A A + c_d d + c_\epsilon \epsilon + c_b b + c_{\bar{w}} \bar{w} + c_\theta \theta \tag{6.7}$$

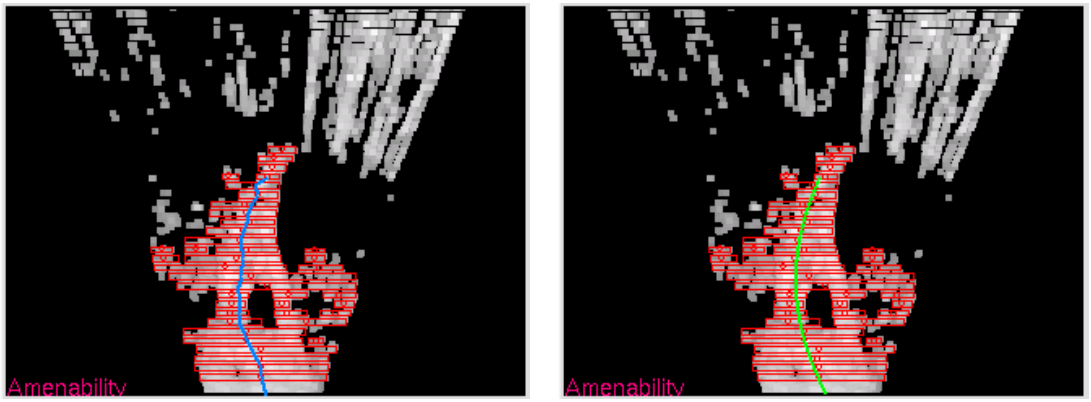
where $\{c_A, c_d, c_\epsilon, c_b, c_{\bar{w}}, c_\theta\}$ are weighting coefficients that describe the relative importance of each trait. Typically we use $c_A = 1.5$, $c_d = 2.0$, $c_\epsilon = -0.5$, $c_b = 1.5$, $c_{\bar{w}} = 1.0$ and $c_\theta = -1.0$. With these values, we favour a highly amenable, long path with large buffering, and weakly penalise paths with a large fitting error, indicating a path that is not smooth, and paths that deviate from the desired bearing. Standard behaviour also simply sets the desired bearing to the previous bearing in order to provide a path that is temporally smooth.

6.5 Experimental Results

In this section a selection of local path-planning results will be presented in order to give an overview of the system's performance.

In the results that follow, the raw centre image, reprojected obstacle likelihood map, reprojected road likelihood map, amenability map (overlaid with segments, graph, trunk and fitted model) and reprojected centre image (overlaid with fitted model) are shown.

Figure 6.11 shows a fairly simple scene where the road veers off to the right, with the only potentially troublesome element being the alternating strong shadows and bright regions. The road map shows, however, that these potentially problematic lighting conditions are handled



(a) Trunk path (b) Fitted model

Figure 6.10: The path trunk

comfortably. There is a patch to the left of the road that is incorrectly considered to be part of the road, however, because of the brightly-lit leaf-litter's resemblance to the road surface. The ground is quite flat and the obstacle map reflects this, though there are some falsely classified obstacle regions in the distance. The amenability map accordingly shows these non-existent obstacles as regions to avoid. However, the overall shape of the safe region on the amenability map is a good representation of the environment, and the resultant path is a smooth curve to the right that follows the centre of the road.

The next result in Figure 6.12 shows another scene with dark shadows and bright regions along the road. The main path veers to the right while a smaller path forks off to the left and, notably, down an incline. The obstacle map identifies the left side of the main road at the fork as being potentially unsafe due to the drop in terrain. The road map clearly shows the left fork in the road though is less clear where the main road continues forwards through a brightly lit region. The amenability map shows the left fork and the region where it joins the main road as being areas to avoid. The path generated veers to the right to avoid these areas.

Figure 6.13 shows a simple scene with little to cause trouble. The road flows slightly to the right and is primarily in shadow: only a few small regions in the foreground are brightly lit. The obstacle map shows that the terrain is essentially flat and hence traversable, while the road map shows the road to have been extracted very well. The amenability map is a similarly accurate representation of the environment and the generated path clearly follows approximate centre of the road.

In Figure 6.14 a large rock is situated on the left side of the road surface, creating a clear obstacle. The obstacle map shows a safe and traversable environment except for the presence of the rock. In spite of there being a fairly small change in appearance between the sandy dirt road and the dry surrounds, the road map shows that the road surface has been very well extracted. The amenability map shows the terrain ahead to be quite safe ahead except for the large rock. The path chosen aims to veer around the rock and to the right while largely staying on the road.

A large rock sits on the right side of an otherwise clear road in Figure 6.15. The road itself gently curves to the left and is vaguely bordered by dry leaf-litter. The road map shows the

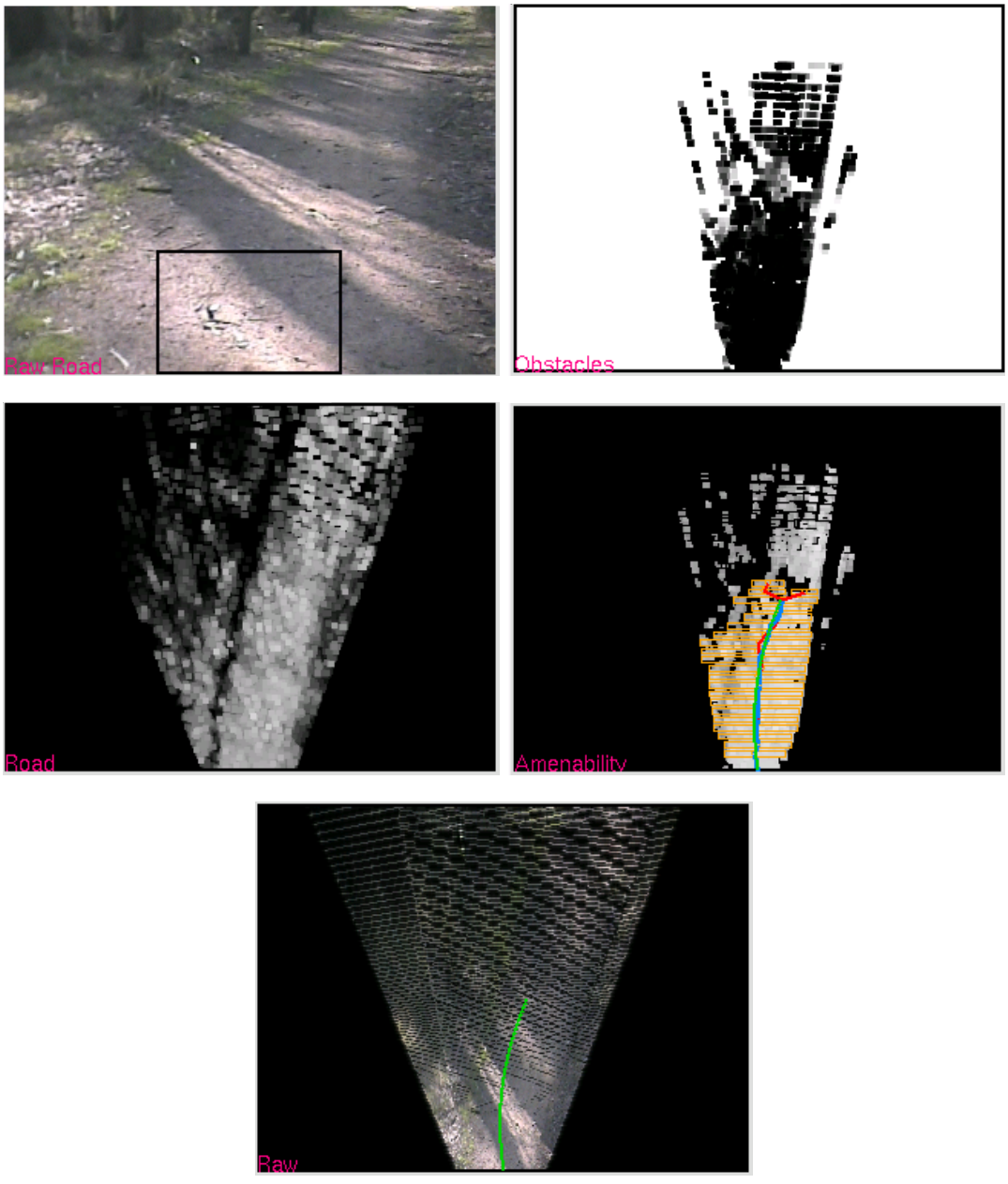


Figure 6.11: Road-following result 1

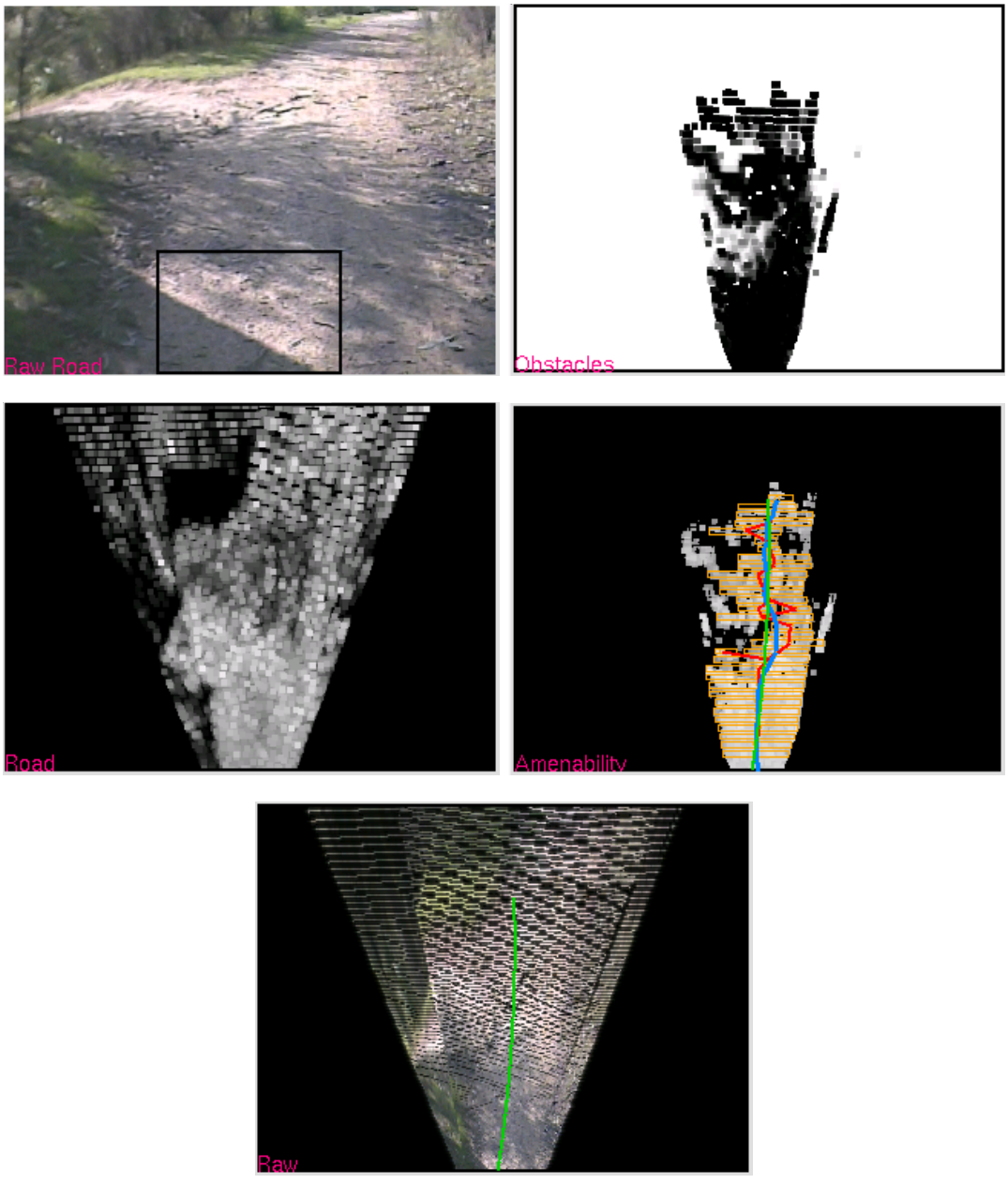


Figure 6.12: Road-following result 2

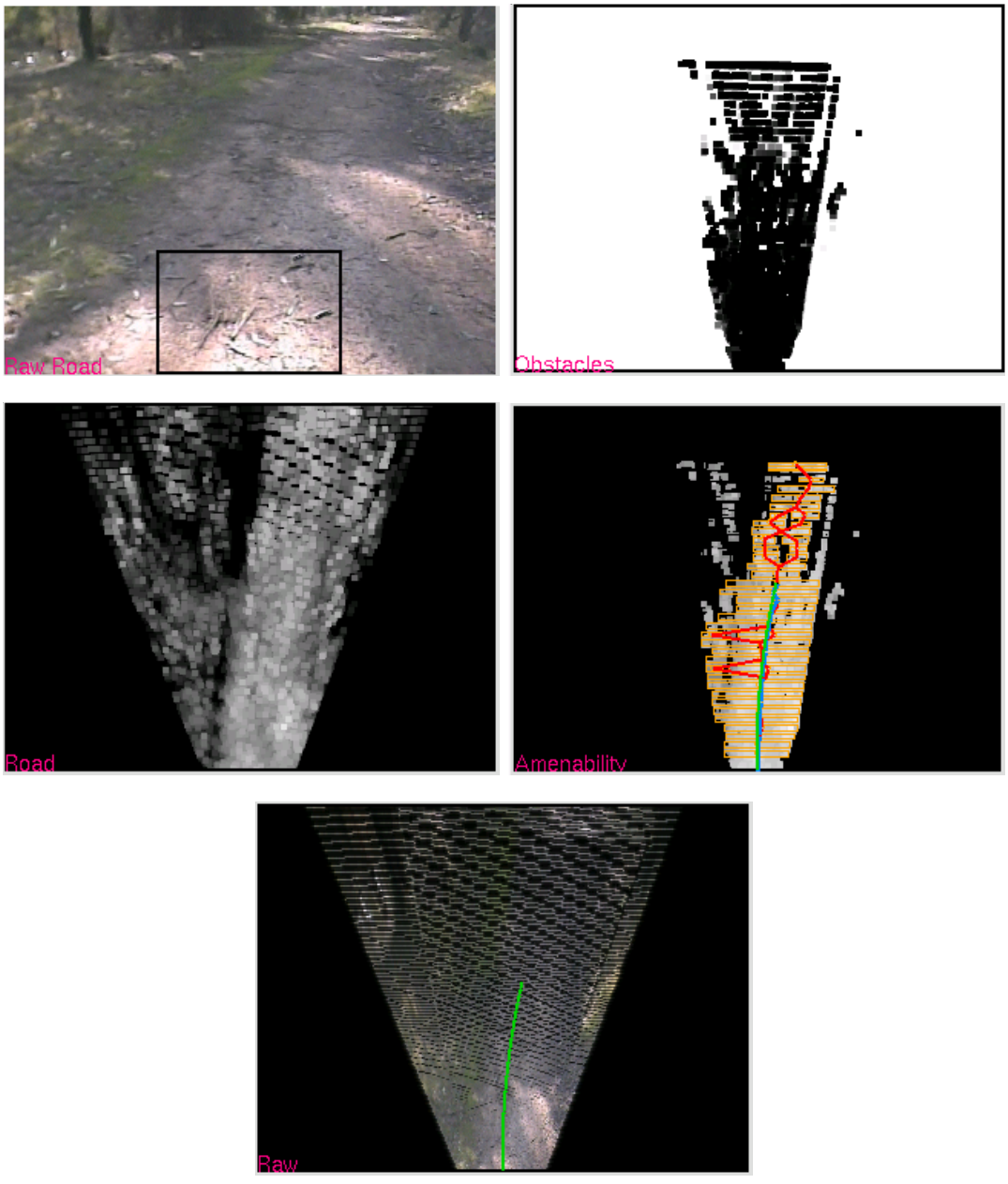


Figure 6.13: Road-following result 3

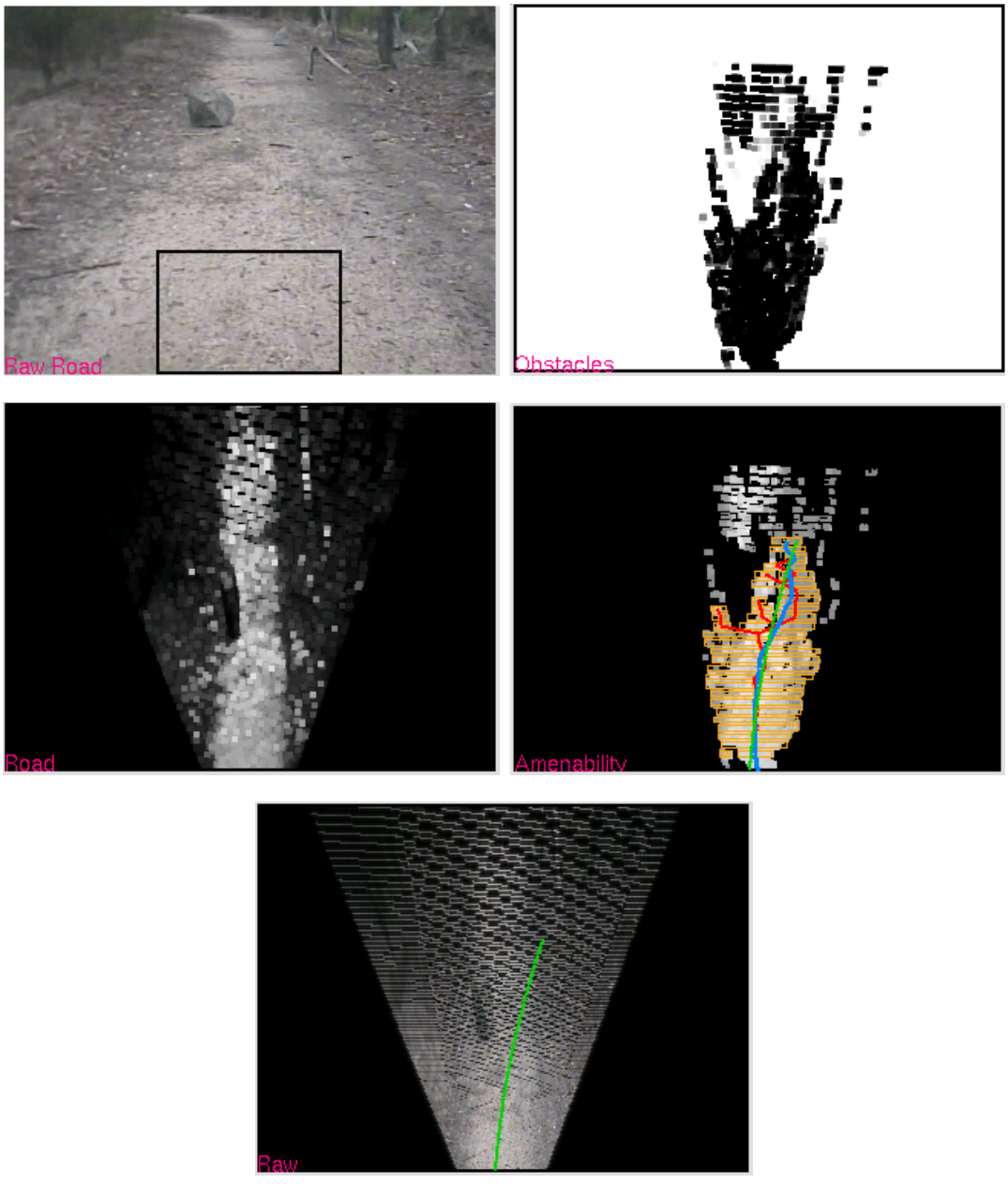


Figure 6.14: Road-following result 4

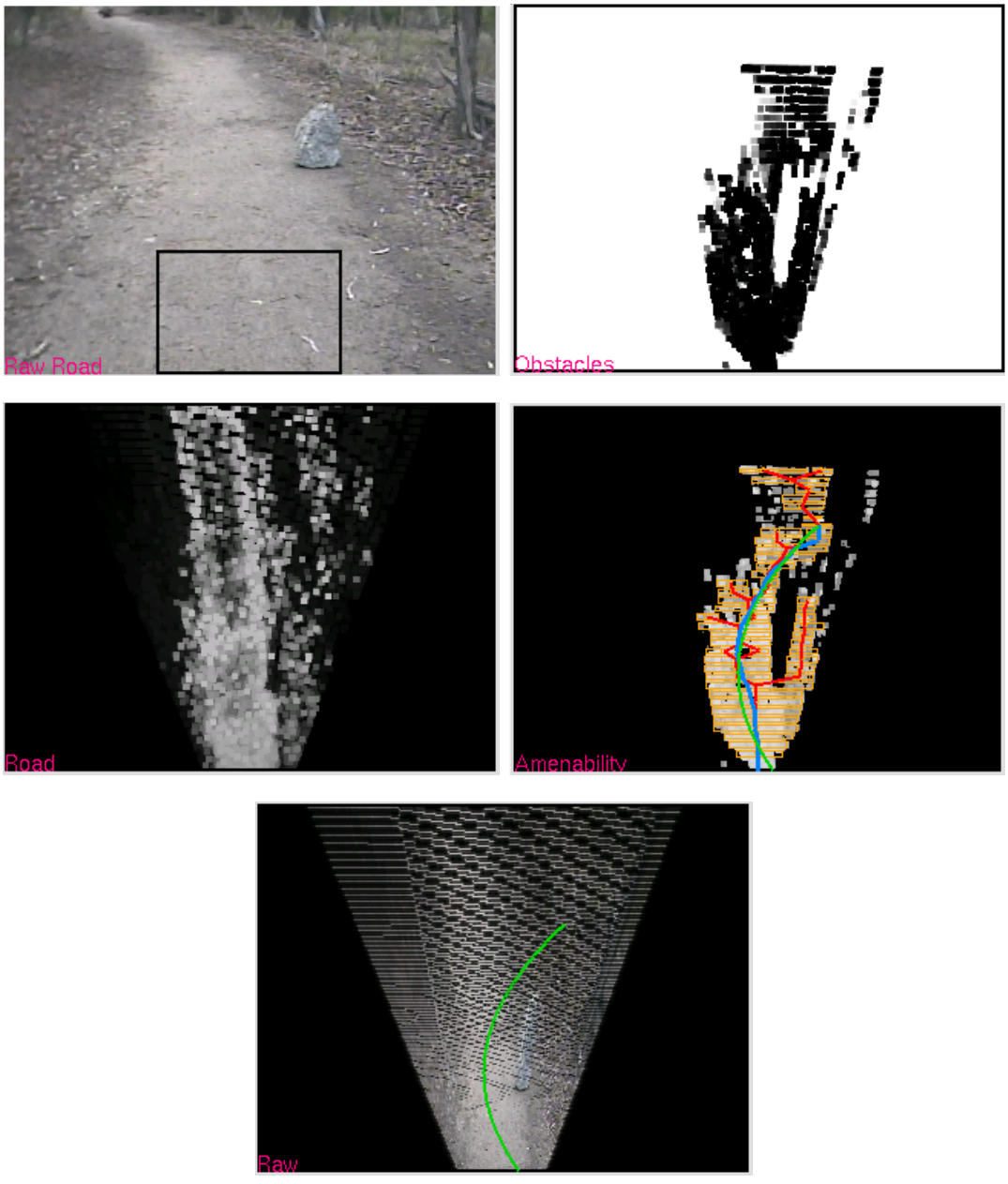


Figure 6.15: Road-following result 5

road to curve to the left, though there is a gap down the centre of the road in the distance, most likely caused by the slightly darker patch of road surface. The obstacle map clearly shows the rock in the middle of otherwise flat and hence safe terrain. There is, however, a misclassified obstacle region on the left beyond the rock. The amenability map closely resembles the obstacle map and as a result of the false-positive obstacle region, the chosen path, while safely avoiding the rock, has veered too far to the right after avoiding the rock and is tending towards the bush area next to the road. However, as the robot progresses first towards and then around the rock, it will move to a position where it is better able to sense and plan an improved path beyond the obstacle.

In Figure 6.16 there is a very subtle obstacle infringing upon an otherwise clear road. There is a feathery overhanging branch that droops from above over the right side of the road. The colour of the branch is very similar to that of parts of the road making it essentially invisible in the road likelihood map (unlike some early obstacles, that had distinctly different appearances to the road). However, it is well detected by the stereo system and appears as a clear obstacle on the obstacle map. The chosen path, while stopping slightly short due to false-positive obstacles in the middle-ground, safely aims to guide the robot around and to the left of the branch, the safer of two alternatives (there is a potential path around and to the right, but the unobstructed road is wider to the left).

In Figure 6.17 there are two distinct obstacles (rocks) on either side of the road. The darker rock on the right is visible on the road map as a non-road region, but the left rock is not. Both are clearly visible on the obstacle map, however. The chosen path guides the robot safely between the obstacles though, again, stops short because of false-positive obstacles in the distance.

Some fallen branches lie on the left of the road in Figure 6.18. They are not at all distinctly visible on the road map, which does otherwise extract the road very clearly. The obstacle map, though, *does* show the presence of obstacles to the left on otherwise flat and clear ground. The generated path veers around and to the right of the branches, stopping short because of sparse stereo information in the distance and small regions of false-positives.

The next set of results show scenes of cross-country traversal. Accordingly, no road likelihood map is shown, and the amenability map is produced using only the obstacle likelihood map.

In Figure 6.5 the robot is facing short, step-like rise in the terrain with the ground quite flat on both sides step but at different levels. The obstacle map clearly shows the higher ground to be unsafe while the foreground and region to the right is flat and so considered safe. The path produced veers to the right, aiming to steer the robot away from the step and guide it parallel to it.

The ground in Figure 6.5 is quite open and flat except for a few small shrubs. The largest shrub in the middle and to the left of the raw image is a clear obstacle and is shown as such on the obstacle map. Many of the other smaller shrubs are also visible as obstacles, though the degradation of stereo with distance (there are typically a number of false-positive regions in the distance) means some of these actual obstacles are not distinctly shown. However, the path produced is clearly seen to avoid both the large shrub to the left-of-centre and also the smaller shrub further away to the right-of-centre.

The scene in Figure 6.5 is quite cluttered on the left with wispy branches and a large tuft of grass. There is a fairly clear path around the shrubbery to the right, and this is firstly shown in the obstacle and amenability maps and secondly taken advantage of by the path-planner.

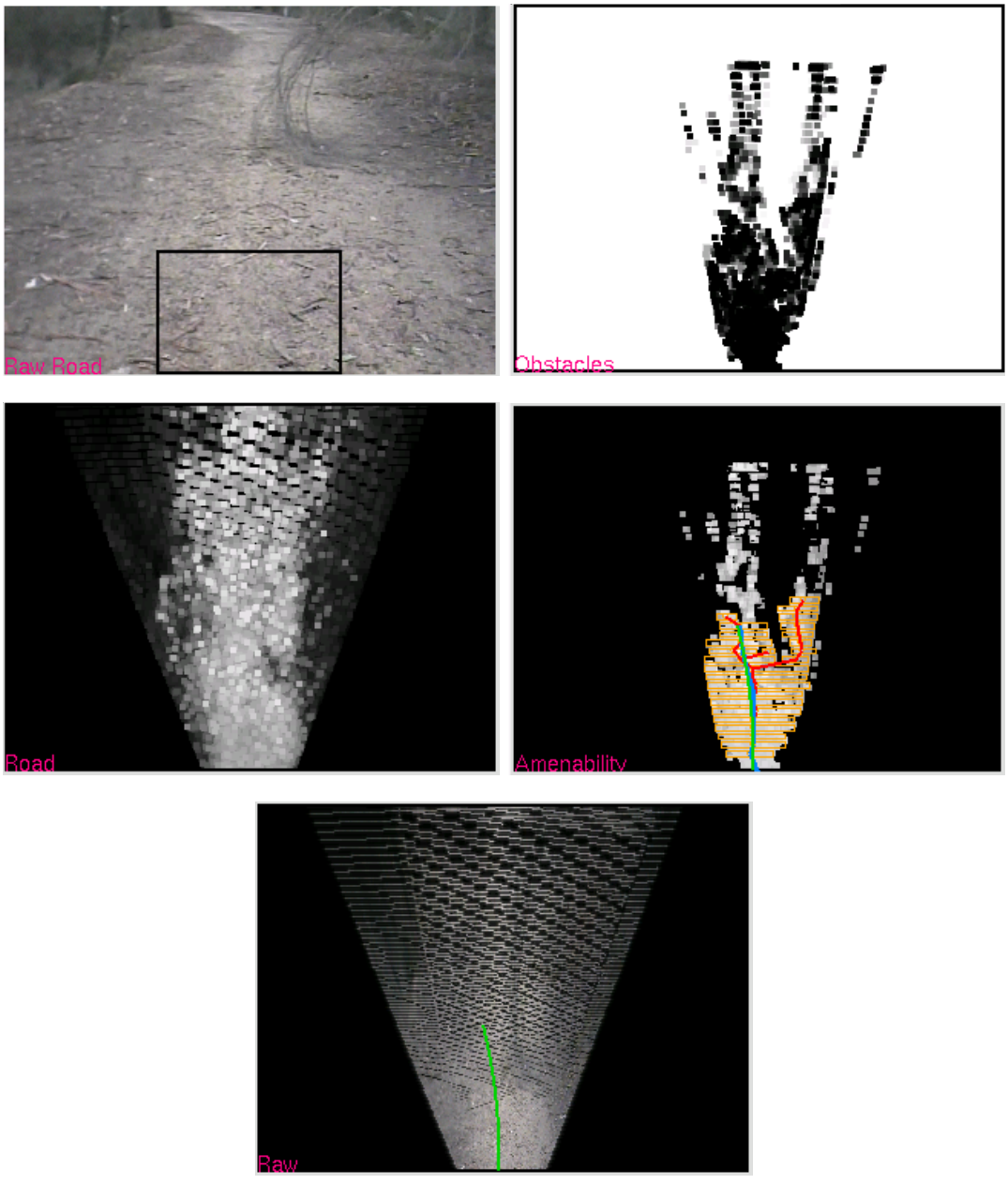


Figure 6.16: Road-following result 6

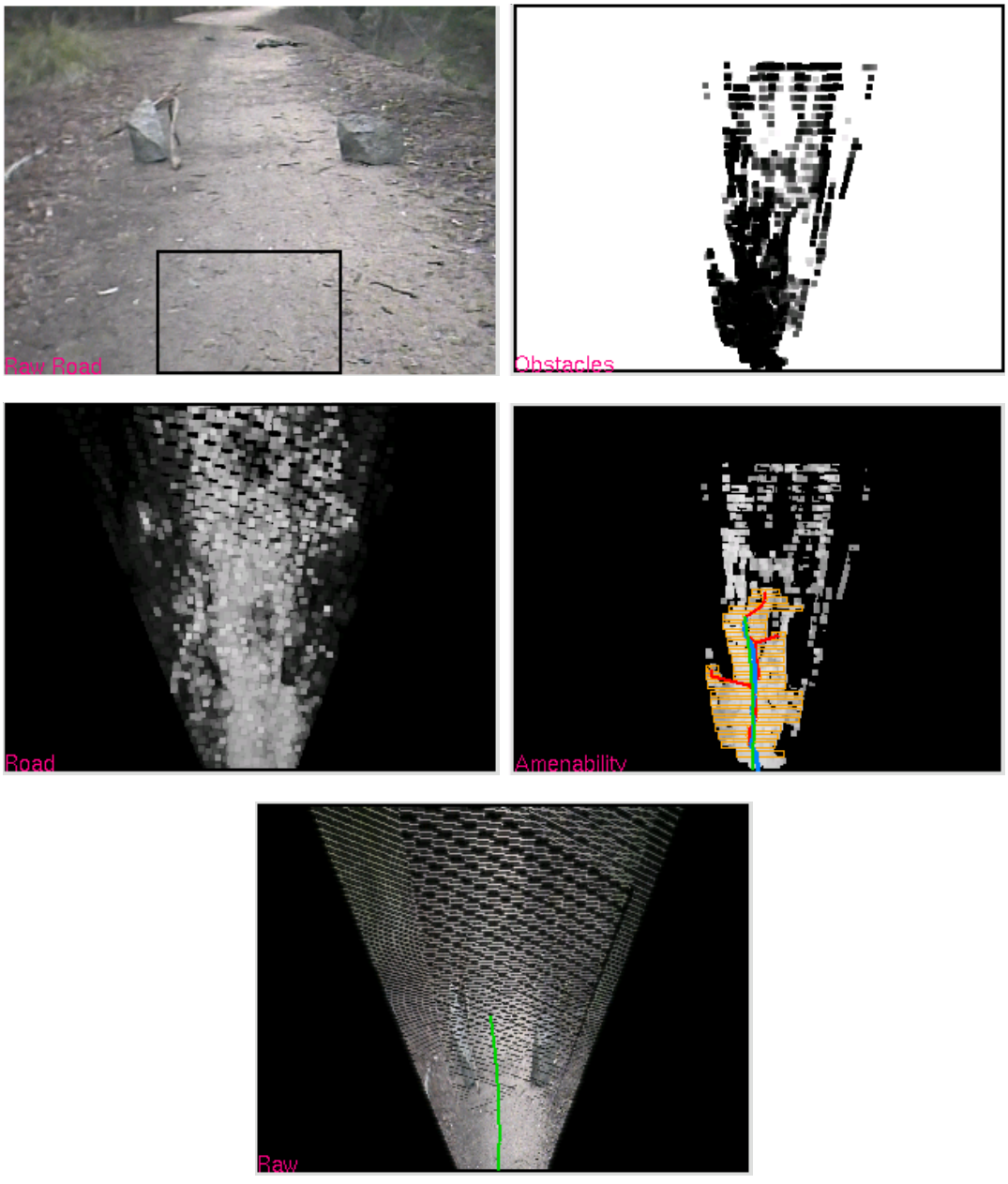


Figure 6.17: Road-following result 7

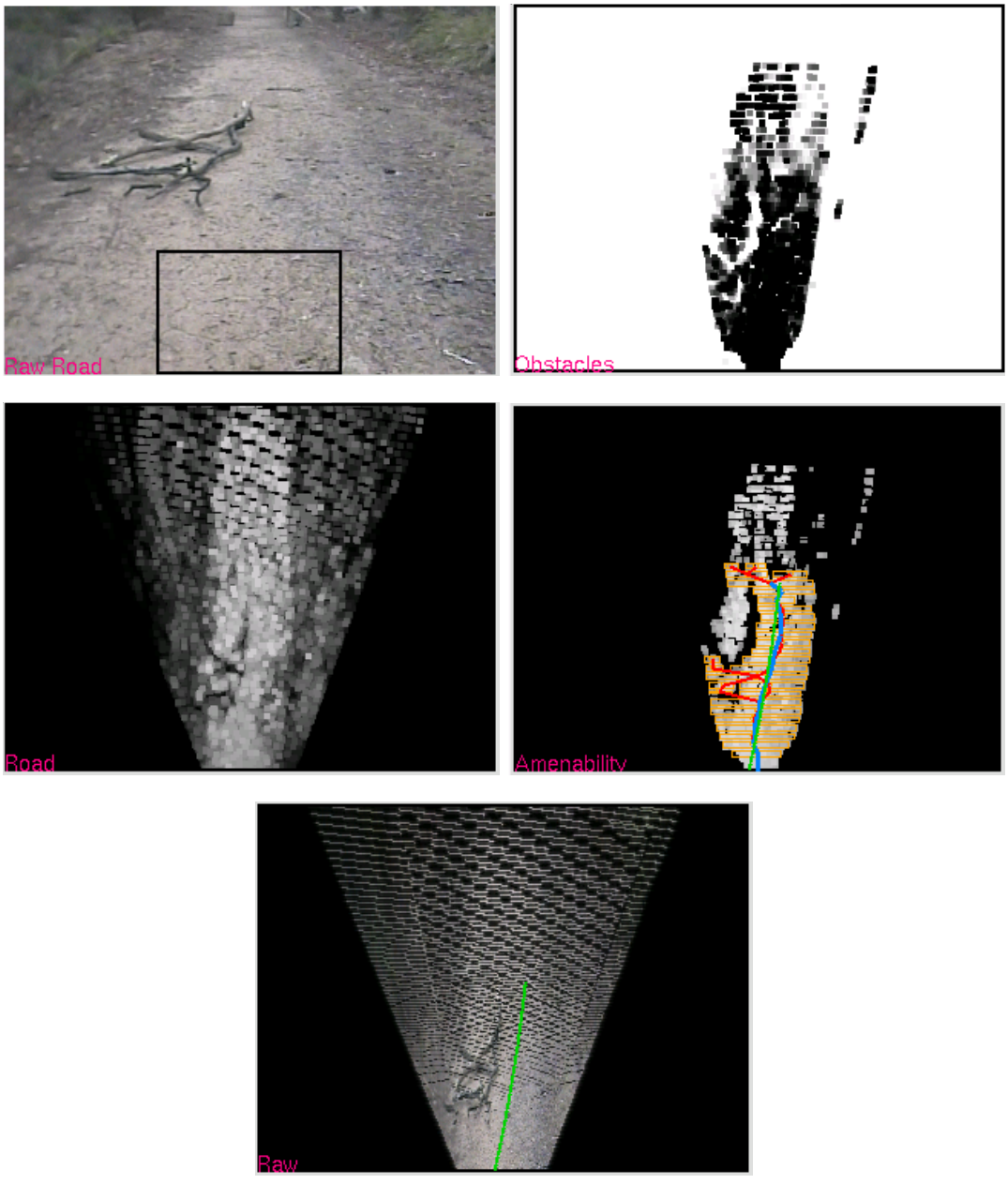


Figure 6.18: Road-following result 8

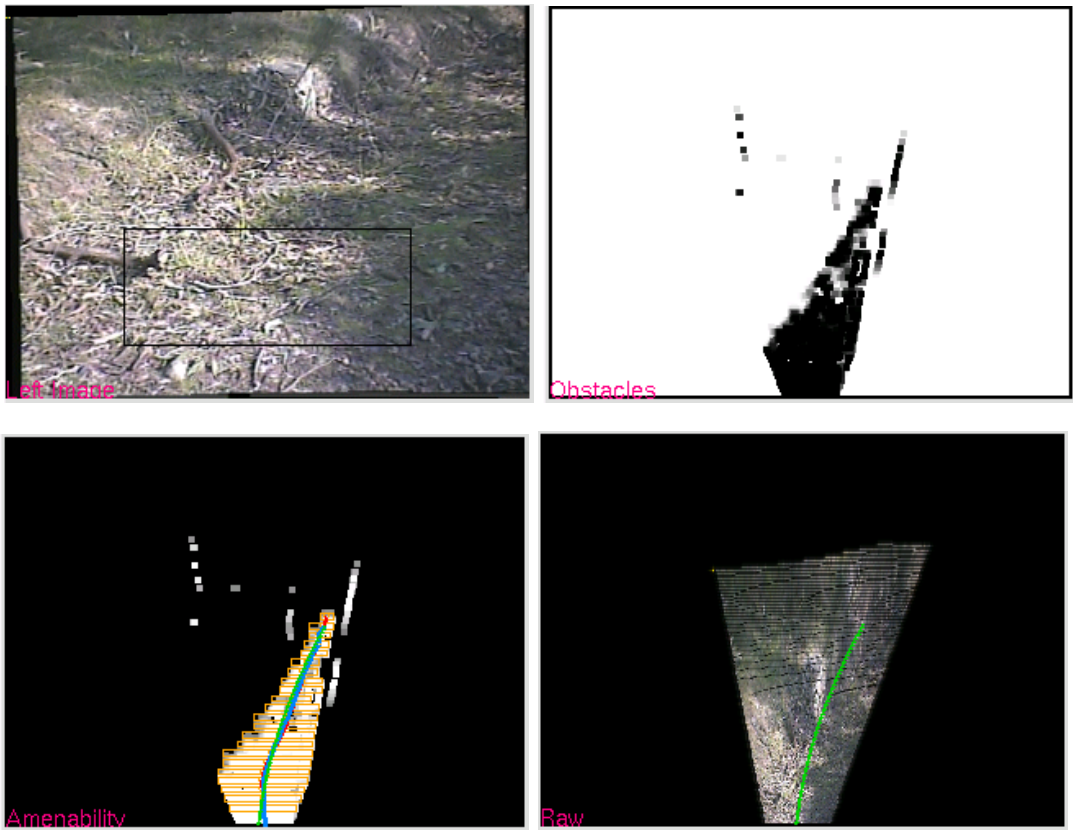


Figure 6.19: Cross-country result 1

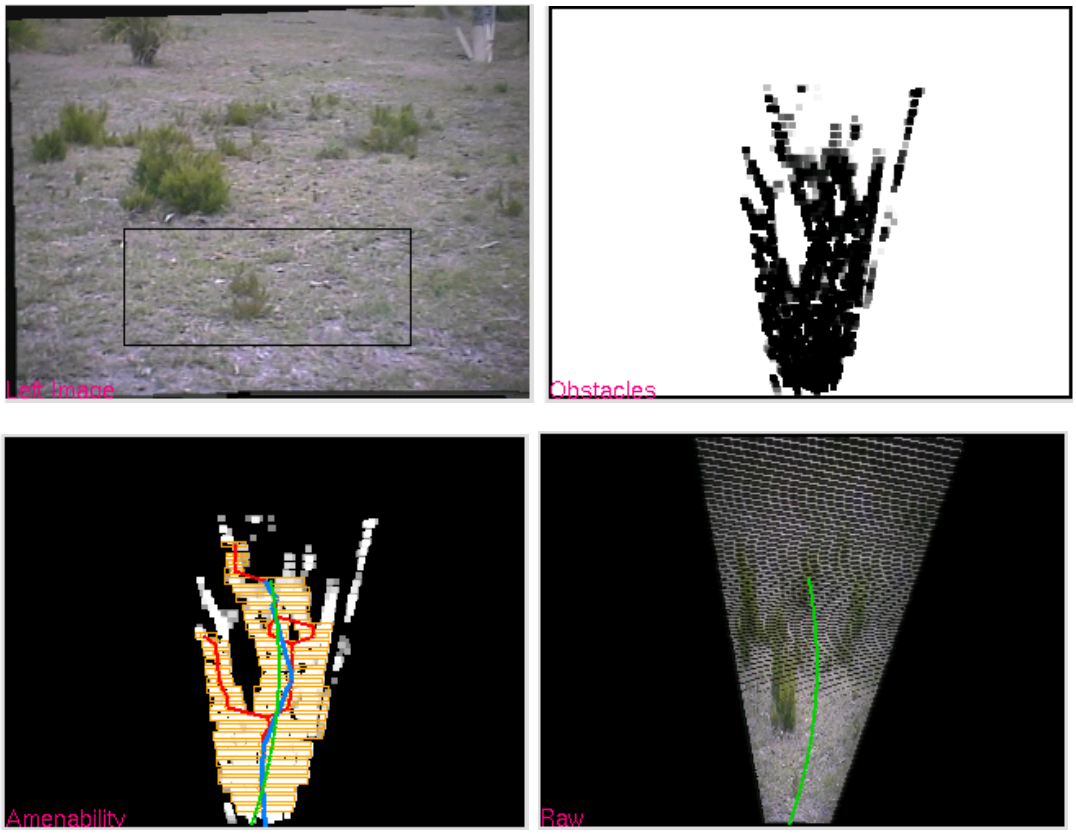


Figure 6.20: Cross-country result 2

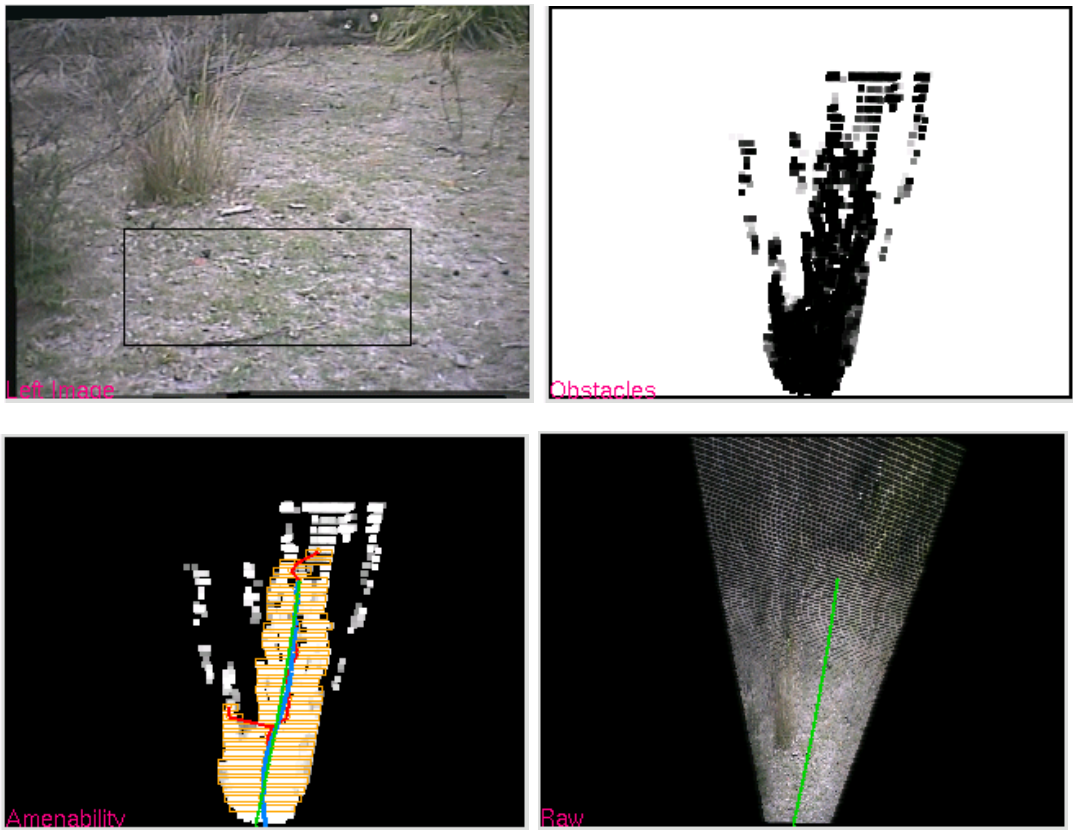


Figure 6.21: Cross-country result 3

A collection of additional experimental results are given on the accompanying CD-ROM, referred to in Appendix .6.

6.6 Conclusion

The problem that this chapter addresses is that of how, given some local sensor data that describes the location of obstacles and the road, can we determine a local path that will guide the robot along the road while avoiding obstacles. Because no specific goal location is given or can be uniquely determined, the process of determining *where* the robot should go is very tightly coupled to the problem of determining *how* the robot should get there.

The suggested local path-planning system that makes use of an obstacle-likelihood map and a road-likelihood map, created by the systems described earlier. Its primary aim is to determine a robot trajectory that follows the intermediate goal of following a dirt road while avoiding obstacles. However, it flexibly allows either feature-map – road-likelihood or obstacle-likelihood – to be ignored, most usefully allows road-likelihood information to be ignored when cross-country traversal is required.

It does this firstly by combining the two likelihood images (when both are required) into a map of amenably traversable terrain that describes, in terms of obstacles and the road, where it is desirable to traverse. Based upon the road-following problem, it then creates a geometric graph describing potential paths that stay within amenable regions. Finally, it evaluates these paths and produces a smooth trajectory that balances covering long distances with remaining safe. In addition, the ability to positively weight potential paths in a given direction gives a means by which, should multiple options present themselves (for example, when facing a fork in the road), a particular choice (directionally speaking) can be made.

The experimental results presented show that the system is able to cope with a wide range of situations where a variety of obstacles – some seen by both road-finder and obstacle-finder, others seen only by the obstacle-finder – are successfully avoided while the path around them stays to the road where possible.

Chapter 7

Global Path-Planning with a Hybrid Map

> “ *The map is not the territory.* ”
> **Alfred Korzybski, 1879–1950**

Figure 7.1: A symbolic and literal view of the terrain, in the form of a road map and aerial photograph. *(© Google Maps Australia [46])*

To navigate at a high level and plan paths between important locations throughout a mission, a robot requires an overall map of the environment. In some circumstances, most notably during exploratory missions, the map will need to be built incrementally and *in situ*.

In other scenarios, it is viable to expect a map to be provided because the terrain is already at least partially known. In either case, in even moderately large-area environments, the way the terrain is represented – that is, the form of the map – can have a strong impact on the methods and techniques available *and* on the complexity and resource requirements for planning paths. A concise and efficient representation of the environment is desired because it will typically have comparatively low resource requirements and will allow paths to be planned relatively quickly. It is important to note, however, that in an uncontrolled outdoor environment like the bush, the environment can change rapidly. Tracks can easily be blocked by fallen trees or heavy rain after a storm, for example, meaning the environment as described by a map is somewhat different from the actual environment as seen during the mission. The ability to adapt and re-plan is invaluable.

7.1 Introduction

The National and State Parks that form the environment of operation, while essentially natural and largely untouched, are not completely unknown. Because of the efforts of local residents, workers, government agencies and private companies, much is known about these environments that can be taken advantage of to assist navigation. Of prime importance here is the knowledge, in the form of existing maps, of the presence and approximate locations of the many tracks that span the environment. A simple example of a map showing such tracks is shown in Figure 7.2. Many commercially available maps (such as the commonly available *Rooftop's Adventure* and *Forest Activities* map series [34]) contain far more detail at a smaller scale, and mapping software based on satellite photography (such as Google Maps [47] and Google Earth [48]) are continuing to improve and provide higher and higher levels of detail.

These tracks represent *"inside information"*: we know beforehand that the probability of having clear passageway between two locations is high – though not guaranteed – if we stay on a track. In addition, because the explicit reason for creating a track is to provide an easily navigable passageway, traversal along tracks is typically faster than an off-road crossing, even when the track distance is greater. In spite of this, traversal off-road may be possible and in some cases preferred over staying on a track. An extreme example of this could occur when a track at the bottom of a valley has become flooded and made impassable. An uphill retreat may be too costly because of a steep and slippery track, so the best alternative would be to find another route off-road.

In the work presented in this chapter, we describe a high-level path-planner intended to utilise the visual-navigation mechanisms presented earlier as navigational behaviours. A full implementation of the planner and integration with the visual mechanisms, however, was unable to be completed due to time constraints. As a result, the ideas presented here are implemented and demonstrated via simulation. Accordingly, it should be stated that the results shown are, regretfully, at best an indication of what may be possible, in acknowledgement of the inadequacy of a simulation to accurately duplicate the reality of navigation in poorly structured environments by a physical robot.

7.2 Requirements

With the stated underlying motivations – the desire to make use of tracks as much as possible while not ruling out cross-country traversal – we must now find a way to represent the

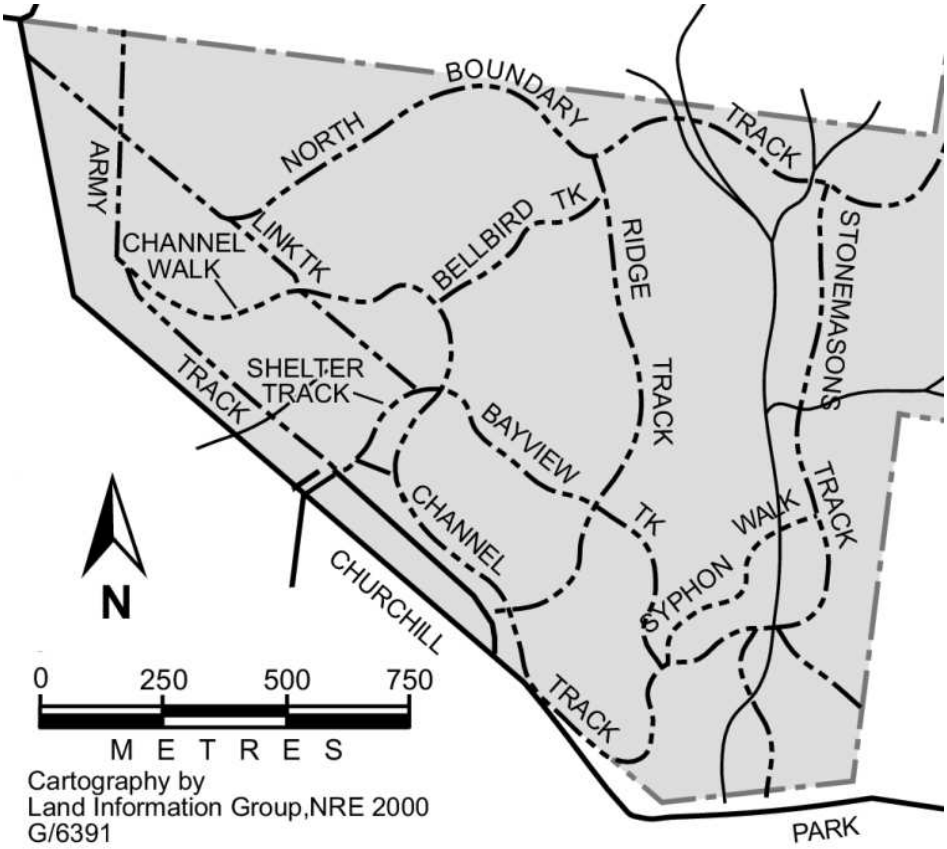


Figure 7.2: A typical bush track-map *(modified from* "Churchill National Park & Lysterfield Park - Visitor Guide" *[106])*

environment that is amenable to these forms of navigation. At this stage we are making the assumption that we have a reasonably accurate track-map of the environment.There are three things we wish to make use of: "points of interest" (typically intersections or simply way-points strategically spaced along long tracks), the existence of tracks between these points, and the lengths of the track between any adjacently connected points.

The task at hand is one of high-level, point-to-point navigation in relatively large-scale natural outdoor environments. We do not need the same precision as would be required if, for comparison, we were navigating a minefield or performing a docking manoeuvre with a space-station. In the most generalised sense we require a mechanism for characterising and recognising a location. In order to avoid getting lost, we need the ability to stop and confirm our location every now and then (in particular, when we reach – or believe we have reached – a point of interest). This will then allow us to continue with our mission if we indeed are where we think we are, or to trigger a corrective behaviour (for example, to continue down the track for a while) if we are not. A number of mechanisms would be able to perform such a duty. A LIDAR unit and associated scan-matching algorithm or a camera coupled with an image-matching algorithm are examples of self-contained solutions. Both of these, however, would require additional prior knowledge, that is, a database of LIDAR scans or images of the points of interest. Such information could be obtained, for example, by manually driving the robot around and obtaining scans or images at appropriate points. However, given that the development an implementation of such a system is well beyond the scope of this work, a more general solution will be sought, one that can be used with the information present in an existing track-map. Additionally, as will be made apparent, the choice made facilitates the development of paths that traverse through unknown terrain which ultimately means a more flexible path-planner.

Given the general sparsity of point-of-interest locations, the freedom to place such points where most useful and the intermittent nature of intended use, GPS is an appropriate choice of sensing. This is in spite of its limitations with regards to signal coverage (see Sections 1.3.2 and 1.4.2), primarily due to satellite line-of-sight interruptions caused by foliage. Without the need for continuous use, we can ensure that points-of-interest are only located where there is a good chance of a clear signal.

We can, with most modern maps, read GPS coordinates directly off the map. If necessary, as with alternatives, we can enhance or augment our existing data with measurements made in the field. For example, the robot could monitor GPS signal strength and add new nodes or fine-tune existing nodes as required. Additionally, the low density of information required means it is not at all unreasonable to suggest that a map of appropriate detail could be created in the field as required. For example, a park ranger could drive along the tracks in a park with a GPS receiver, logging his or her movement both using the GPS unit and the vehicle's odometer.

At this point it must be noted that the material in this chapter deviates from the theme of navigation using passive vision alone. Many of the ideas presented here could be implemented using passive vision systems (as noted above), though it is believed that a simpler, more general and overall better solution will be reached through the use of alternative primary sensing as stated.

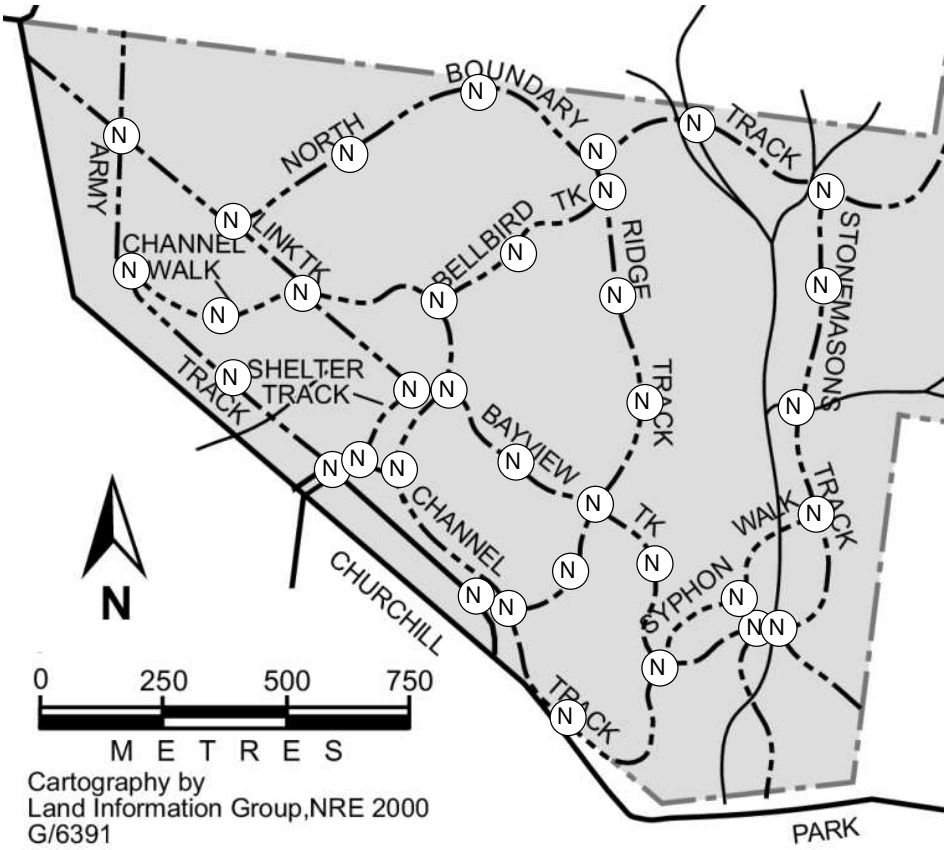


Figure 7.3: Track-map, overlaid with nodes *(modified from* "Churchill National Park & Lysterfield Park - Visitor Guide" *[106])*

7.3 Data Representation

7.3.1 Metric Locations

The first aspect we wish to represent is a point of interest, in particular, given our choice of sensing, the location of such a point. These points are referred to as nodes and contain a metric description – that is, a coordinate – of a geographic position. Nodes with be represented symbolically as **N**.

Figure 7.3 shows the same map in Figure 7.2 with overlaid nodes, each shown as a circled **N**. In this figure, the nodes are what will be referred to as *real* nodes, as they correspond to *actual* points-of-interest, a distinction that will become clearer in the following sections.

7.3.2 Topological Tracks

The second aspect we wish to represent is a segment of a track that connects two nodes. These are referred to as edges and contain a length and departure bearing (metric measurements) and a per-unit-length weight (a real number that may or may not have a relationship with the environment). An edge will be represented symbolically as **E**. Edges are directional with the primary motivation being to allow for direction-based costs, that is, situations where it

is more expensive to traverse one way along an edge than the other. An example of this is a steep track where it is less costly in terms of energy usage to travel down that it is to travel up. Another more extreme example is that of one-way roads, most commonly found in urban environments but also present in some areas of parks, such as carparks and park entrances. While it would typically be physically possible to travel in both directions, the risk of going the "*wrong way*" – and potentially being faced with another vehicle travelling towards the robot – means the cost of traversal is dependent on the direction of travel.

Edge lengths serve two purposes: firstly, they are used to determine a *relative* position along a track, and secondly, when multiplied by the edge weight, they give an edge cost which is used to compare edges when searching for an overall best path. The length measurement is intended to be used in combination with odometry to determine a relative position along a track segment, in order to trigger a localisation operation. For example, knowing that the current segment is one hundred metres long and that the robot has travelled one hundred metres along the track from the last-visited node, we can determine that we should be at (or very near to) the destination node and should localise to confirm this.

An edge's departure bearing is a practical requirement. It provides a means for discerning between multiple edges that depart from a node. A bearing has three main advantages: Firstly, it can be deduced *a priori* from a track-map of sufficient detail (or again it could be refined from field measurements). Secondly, the only requirement is a compass, a long-used navigation tool that requires no interpretation or processing of data to be immediately useful. A compass is also likely to be very reliable in the bush where there is little chance of interference caused by large metal structures or power-lines or other similar common sources of compass-error. Thirdly, unlike simple edge enumeration (for example, clockwise numbering of the edges from a reference edge), there is no room for ambiguity. As an example, an edge departing at a due-north bearing is more clearly defined than one defined to be the second clockwise edge, as to find the enumerated edge we must first either find the reference edge or determine the arrival edge, both of which are susceptible to error. Finally, as will be made evident, it is a convenient parameter in the generation of cross-country paths.

While the edge length and departure bearing are metric measurements, the edge as a whole is *not* a strongly metric representation of the environment. The edge does not take into account the shape of the track or any track geometry (other than the aforementioned length). It is a descriptor of the connectedness of locations and an encapsulation of how (in behavioural terms) travel between those locations can be achieved, that is, by following a track or taking a straight-line, cross-country route.

7.3.3 Pretend Positions?

In order to accommodate directed, cross-country traversal, a further node type will be introduced. *Pseudo-nodes*, like real nodes, correspond to metric, physical locations. Unlike real nodes, they do not correspond to any *distinct* location or environmental feature. Pseudo-nodes are nodes that are placed arbitrarily – but not intentionally on tracks – and provide a means for creating paths cross-country. Pseudo-nodes are placed uniformly on a relatively coarse grid throughout the map.

As stated, nodes contain a finely-grained geographic position. With the inclusion of initially unconnected pseudo-nodes, in order to facilitate an efficient search, nodes are augmented with a grid-reference, that is, a coordinate of coarser grain. For distinction, the finely-grained position will be referred to simply as a *coordinate*, while the position within the coarse grid

will be described as a *reference*. It should be noted that coordinates are unique: no two nodes may have the same coordinates. Alternatively stated, any two nodes having the same coordinates are in fact the same node. References, however, are not unique. Many nodes may share the same reference, and the only inference that may be made about two nodes with the same reference is that they are located within some bounded distance of each-other.

A pseudo-node can be connected to any immediately adjacent node (or *all* immediately adjacent nodes), forming a *pseudo-edge*, representing a two-way straight-line path of a fixed length at a given bearing.

Knowing the metric location of every node – both real and pseudo – and with a pseudo-edge being a straight-line path, the length and bearing of the pseudo-edge can be determined without additional information. As a pseudo-edge represents cross-country traversal across unknown terrain, pseudo-edges have high weights associated with them. This discourages their use but still allows for off-road travel where there would be significant savings (that is, the perceived cost of risking a cross-country path is less than the cost of taking a much longer but potentially safer track-based path).

7.4 A Hybrid Metric-Topological Map

Edges are inherently topological descriptors. They encapsulate a parameterised behaviour describing how travel between two distinct places can be achieved. In this case, two such behaviours exist: real edges correspond to following a track for a given distance while pseudo-edges correspond to traversal in a straight line at a given bearing for a given distance. Parameters are the edge type (defining the behaviour), an edge length and departure bearing – physical quantities – and a weight, a predetermined value that describes the per-unit-length expense of traversal along that edge. It should also be noted that, while local and immediate information is gathered and used *in situ* to perform the behaviour, no other *a priori* information is required to complete the operation in either case.

Without any other knowledge, we can simply assign a higher weight to pseudo-edges than to real edges, reflecting the risk of traversing off-road compared to the safety of staying on-road. Further knowledge can allow more varied weights. For example, edges corresponding to tracks can be given weights that reflect the condition of the track (that is, how easily traversed it is), the form of terrain in general (remembering that edges are directed a downhill edge would be less costly than an uphill edge) or expected traffic (with busy tracks being more costly). In general, a set of component weights can be used, each describing a particular property. The overall weight will then be the sum of the components. Additionally, weights need not be static properties. As a robot traverses it may learn more about the environment and accordingly update the weights. A simple example would see the robot altering the weight based upon a comparison of the time taken to traverse an edge against a time predicted by the edge length and an assumed average speed.

In the framework presented here, nodes are inherently metric entities with a very strong geometric relationship existing between nodes. They are defined by their position in terms of a coordinate and a reference. This is primarily a result of the introduction of pseudo-nodes, as they represent locations about which nothing other than geographic location is known. Effective navigation in the target environment means maintaining the ability to stray off tracks, whether through desire or necessity. This means going to places about which the robot has very little knowledge. Pseudo-nodes and pseudo-edges provide a means to

Node: **N**	
name	n
type	$t \in \{real, pseudo\}$
coordinate	(x, y)
reference	(row, col)
arrivals	$\{\mathbf{E} : \mathbf{E}\texttt{[destination]} = n\}$
departures	$\{\mathbf{E} : \mathbf{E}\texttt{[source]} = n\}$

(a) Node data structure

Edge: **E**	
name	n
type	$t \in \{real, pseudo\}$
source	$\mathbf{N}_{src}$
bearing	b
destination	$\mathbf{N}_{dst}$
length	l
weight	$w = \sum\{w_0, \ldots, w_i\}$
status	$s \in \{unknown, clear, blocked\}$

(b) Edge data structure

Figure 7.4: Basic map element data structures

accomplish this while – as much as possible – maintaining the underlying and simplifying topological structure of an environment containing many tracks. The result is a Hybrid Metric-Topological map, one that combines a grid-based metric representation of locations with a graph-based topological representation of how one can move from place to place.

7.5 Representing a Hybrid Map

At the lowest level, two basic data structures are required to store nodes and edges. Nodes, shown in Figure 7.4(a), contain a name, type, coordinate, reference, and two sets of edges, one for edges that arrive at this node, one for edges that depart from this node.

Edges, shown in Figure 7.4(b), contain a name, type, source node, departure bearing, destination node, length, weight and a status. The weight is in general the sum of multiple components. The total cost of traversing an edge is therefore the length multiplied by the weight. As a simple example, a real edge with a length of 123 metres and a weight of 1.5 would have a cost of 184.5. A shorter, but more heavily weighted pseudo edge with a length of 80 and a weight of 2.4 would have a greater cost of 192. The status, one of *unknown*, *clear* and *blocked*, is used to describe the known state of an edge in terms of traversability. Initially, all edges are given a status of *unknown*, which is effectively the same as *"assumed clear"*. Both *unknown* and *clear* edges are considered for inclusion in a generated path, while *blocked* edges are excluded. The status may be changed when new information is gained. For example, an initially *unknown* edge will be given a status of *clear* upon successful traversal, while unsuccessful traversal, because a fallen tree had blocked the track, for example, will result in status of *blocked*.

The inclusion of source and destination nodes within an edge, and lists of arriving and

Grid Cell: **C**	
origin	(x, y)
reference	(row, col)
nodes	$\{\mathbf{N} : \mathbf{N}[\texttt{reference}] = (row, col)\}$

(a) Grid cell data structure

Grid: **G**	
grid size	$(rows, cols)$
cell size	$(\Delta x_{cell}, \Delta y_{cell})$
origin	(x, y)
cells	$\{\mathbf{C}\}$

(b) Grid data structure

Figure 7.5: Map grid data structures

departing edges within a node provide the association necessary for the formation of a topological graph. With a list of nodes and a list of edges, we can select any desired pair of start and goal node and perform a search to find a path between them (if such a path exists).

To accommodate pseudo-nodes and pseudo-edges, we could very easily simply overlay our map with a grid, populate the grid with pseudo-nodes, and then connect each pseudo-node to all neighbouring nodes, *a priori*. We would then have a highly-connected graph with all the information we need to allow for track-based or cross-country path planning. This would lead to an increase in both storage requirements and processing required to find a path. It therefore, at least in part, reduces some of the great benefits of a topological versus a metric map: conciseness and efficiency. Such a solution would not scale well either. With each internal pseudo-node (that is, pseudo-nodes not on the border of the map) being connected to at least eight other nodes (as each neighbouring grid-cell will in the least contain one pseudo-node, and potentially multiple real nodes), each pseudo-node introduces at least sixteen edges (as edges are directed). A map fully populated with pseudo-nodes adds a significant number of edges to the graph and, with the underlying goal being to stay on tracks when possible, most of these will never be used. A more efficient solution is to dynamically create pseudo-nodes and their corresponding edges as required.

In order to generate pseudo-nodes dynamically, we first need to know *where* to create them. The use of a map grid and inclusion of references in nodes allows this. Whenever a path needs to be found, we can define an effective region that is simply a rectangle, whose opposite corners are the references of the start and goal nodes. We then populate this region with pseudo-nodes and generate corresponding pseudo-edges. Without further knowledge, and within the constraints of the pseudo-node framework, we are highly unlikely to find a cross-country path better than one contained in such a region. It should be noted that a cross-country path that makes use of intermediate pseudo-nodes is unlikely to contain the most direct cross-country segments between real nodes. This is a result of the grid-based nature of pseudo-node positioning which means a cross-country path between real nodes, via one or more pseudo-nodes, will invariably *not* be a straight-line, but instead will deviate through the pseudo-node(s).

Our final requirement for the hybrid map is a grid structure. Each grid cell will contain a list of the nodes that lie within that cell, providing an easy and efficient way to find neighbouring nodes when connecting pseudo-nodes. With a grid in place, we can simply connect a pseudo-node with all real nodes that are in the same cell or in any of the eight surrounding cells. The need to actually search for nearby nodes is hence removed.

Figure 7.5(a) shows the data-structure used to represent a grid cell. It contains an origin (the coordinate of the south-west corner of that cell), a reference and a set of nodes whose

Map: **M**	
nodes	{**N**}
edges	{**E**}
grid	**G**

Figure 7.6: Overall map data structure

references are the same as that of the grid cell itself. Shown in Figure 7.5(b) is the data-structure of the overall grid. It contains the grid size (the number of rows and columns), the cell size (in coordinate units, describing the area that a single cell covers), an origin (the coordinate of the most south-west corner of the grid) and a set of all grid cells.

Figure 7.6 shows the overall map data-structure. It simply contains a set of nodes, a set of edges and a grid.

7.6 Map Creation

There are only three essential steps in the creation of an initial map:

1. Create a grid to cover the track-map
2. Place real nodes at points of interest, inserting them into the set of nodes and the appropriate grid-cell
3. Connect real nodes with edges as appropriate, inserting the edges into the set of edges

Steps two and three are shown graphically in Figure 7.7 and Figure 7.8, applied to the track-map shown in Figure 7.2.

These steps must be performed manually, although it is conceivable that this could be automated at least partially with the use of additional image-processing software. Even so, a human operator would most likely still be required to check the resultant map, if only to ensure nodes are placed at useful locations. The automation of this procedure is, however, beyond the scope of this work. The simulator implemented has the facility to easily create such maps, using an image of a track-map as a manual guide if desired. A simple point-and-click interface allows the user to place nodes and create edges wherever desired. The type and coordinate of nodes and the type, length, weight and status of edges can also easily be modified. Grid cell size can also be easily set. Additionally, the user may create edge blockages that simulate an initially unseen track blockage that would cause a backtrack and re-plan. Further sections assume that a map has already been created by hand.

7.7 Path Creation

Beginning with given source and destination nodes $\mathbf{N}_s$ and $\mathbf{N}_d$, path creation has four basic steps:

1. Determine the active region from $\mathbf{N}_s$ and $\mathbf{N}_d$
2. Place a pseudo-node in each grid cell within the active region

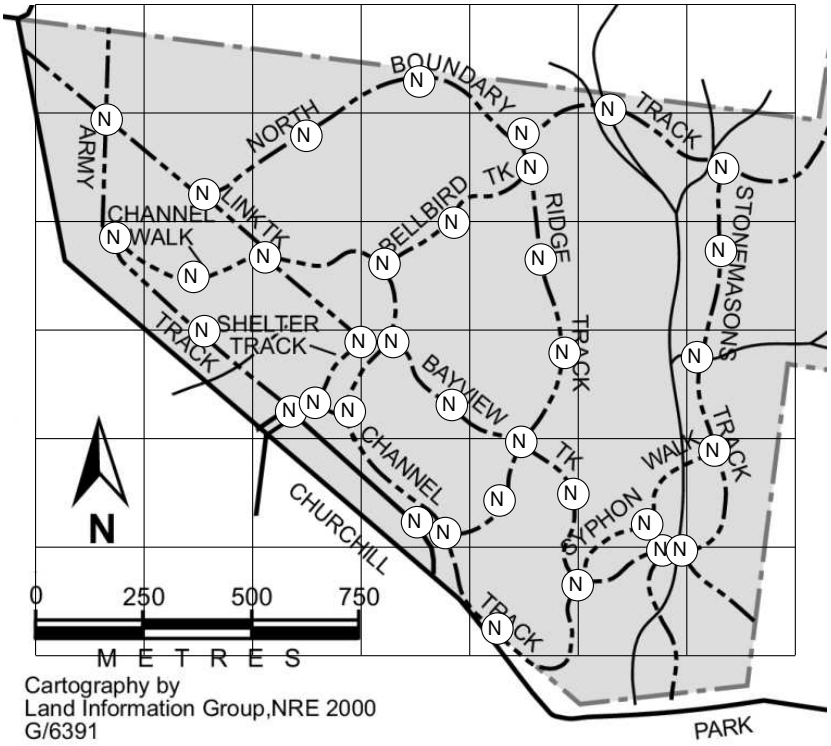


Figure 7.7: Track-map overlaid with grid and real nodes

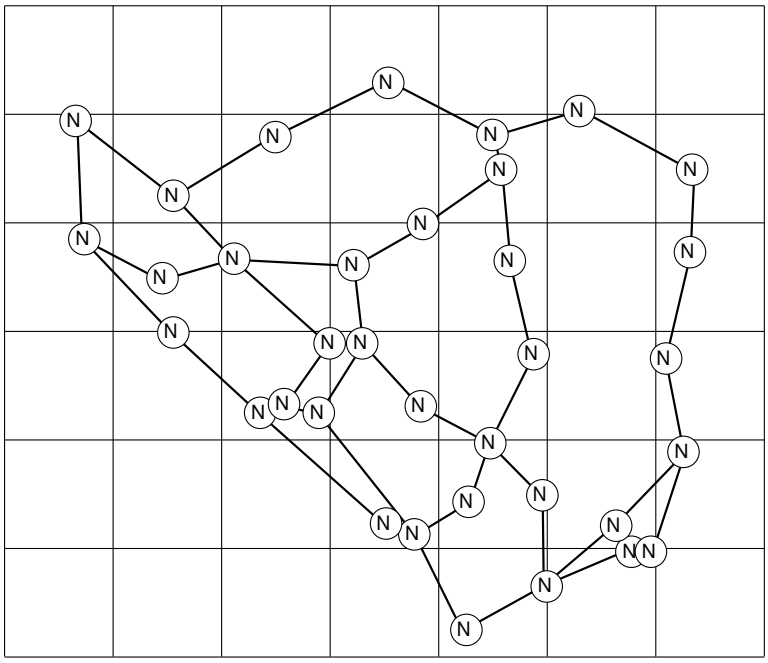


Figure 7.8: Hybrid grid-graph representation

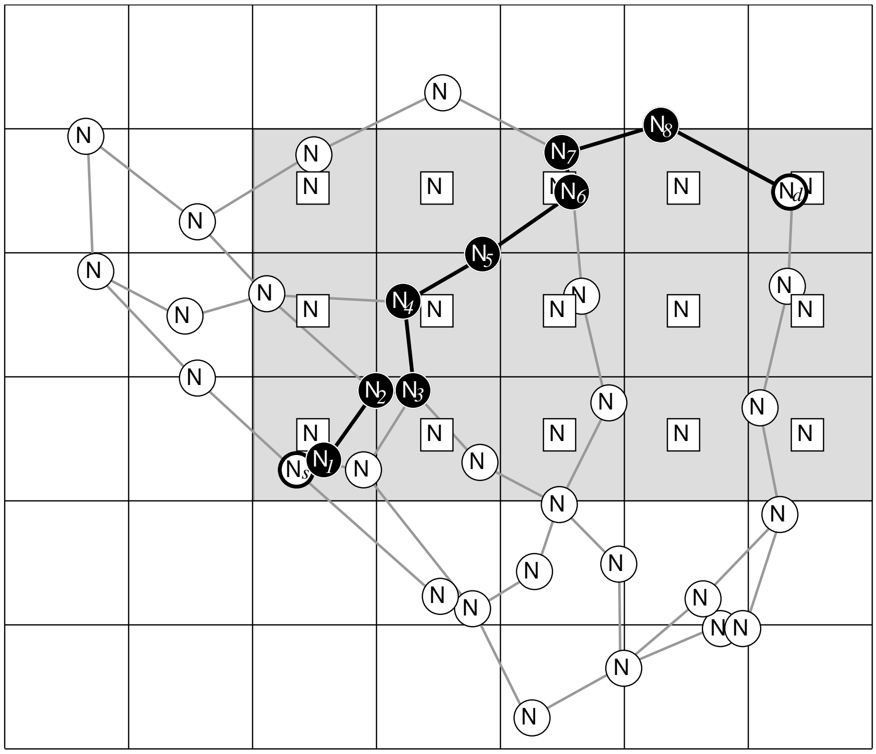


Figure 7.9: Initial path from $\mathbf{N}_s$ to $\mathbf{N}_d$

3. Connect each pseudo-node to every real and pseudo-node in the same cell and in the eight surrounding cells

4. Find the least costly path between $\mathbf{N}_s$ and $\mathbf{N}_d$

Any standard graph-based shortest-path algorithm can be used for step four. Because the process of planning a high-level path will only be performed intermittently, the particular algorithm chosen, in terms of efficiency or speed, is not very important. As such, the standard and well-proven Dijkstra's single-source shortest path algorithm (detailed, for example in [21]) is used with an appropriate termination condition such that processing stops when a path to $\mathbf{N}_d$ is found. The algorithm is modified slightly to accommodate an edge's status such that edges with a *blocked* status are ignored (from a path-creation perspective, blocked edges do not exist).

Figure 7.9 shows the path created to traverse from $\mathbf{N}_s$ to $\mathbf{N}_d$. (To assist clarity, pseudo-edges are not shown in this set of figures.) The *"active region"*, in which pseudo-nodes are generated, is shown as a shaded rectangle, spanning from $\mathbf{N}_s$ at the bottom left to $\mathbf{N}_d$ at the top right. An inserted pseudo-node is depicted as a $\mathbf{N}$ enclosed in a square. Nodes included in the generated path are shown as darkly shaded nodes. Additionally, the path-nodes are numbered 1 through 8 and the edges included in the path are depicted as strong, dark lines.

7.8 Navigation

The general method of navigation, assuming the robot is given a destination node $\mathbf{N}_d$, is shown in Algorithm 7.1. After performing a localisation operation to determine the starting location, a path is found from this source node to the desired destination node. We then iteratively retrieve the next edge and corresponding intermediate goal node from the path, and perform either a road-following or cross-country traversal operation, depending the type of edge. After travelling for the distance specified by the edge, we localise again and determine how far from the goal node we are. If we are not at the goal node, the road-following or cross-country trekking continues until we reach the goal, otherwise we note that the edge was clear and we select the next edge and corresponding goal node. This continues until the current node is the same as the destination node. If, during either following the road or trekking cross-country, we encounter a blockage which prevents further travel, we backtrack to the last node, note the edge as being blocked, and find a new path to the destination. The main motivation for backtracking to the last-visited node, as opposed to re-planning from the current location, is to ensure that the robot is at a clear, known location before re-planning. Because the only knowledge we have of roads is encompassed in edges, which contain only their endpoints and length – and no additional geometry – a re-plan from part-way down a road could easily result in a greedy – and potentially unsafe – attempt to trek cross-country. We choose here to instead err on the side of caution.

Following from the initial path shown in Figure 7.9, in the scenario shown in Figure 7.10, a track blockage is encountered along the edge connecting $\mathbf{N}_3$ and what was in Figure 7.9 $\mathbf{N}_4$. This blockage is depicted as a dotted edge with a strike through it. The section of the path already traversed (up to $\mathbf{N}_3$) is shown as dashed lines. This blockage forces a re-plan. The new effective start node is $\mathbf{N}_3$, so the new active region begins in the cell housing $\mathbf{N}_3$. The new path creates a detour, but still makes use of only real edges.

Figure 7.11 shows a second blockage being encountered, between $\mathbf{N}_7$ and $\mathbf{N}_8$ in Figure 7.10 (again depicted as a dotted path with a strike). Again, a re-plan is necessary and a new active region is created. This time, it is deemed that a relatively short cross-country trek (via pseudo-node $\mathbf{N}_8$) is more efficient than a long, track-based detour.

7.9 Simulation Results

In this section we present some simulated results that show the path-planner in operation, highlighting the use of pseudo-nodes when cross-country trekking is deemed beneficial or necessary.

Figure 7.12 shows the example map that will be used for the next few path-planning examples. Unless stated otherwise, all edges are assumed traversable, that is, no edges are blocked. Additionally, all real edges (shown in Figure 7.12) have edge weights of 1.0, while dynamically created pseudo-edges are created with an edge weight of 1.5, following the assumption that cross-country traversal is more costly than on-road traversal. Edges are shown as half green and half red, with the green half connecting to the edge's source node and the red half connecting to the edge's destination node, noting that edges are directional.

The path shown in Figure 7.13, while somewhat trivial, serves to illustrate the appearance of paths in subsequent results. Here the source node is the left-most node (shown in green), the destination node is the top-most node (shown in red) and the path created between them

Algorithm 7.1: Hybrid-map Navigation

Procedure: `Navigate(`$\mathbf{N}_d$`, M)`

Input: destination node $\mathbf{N}_d$
Input: map $\mathbf{M}$

Data: source node $\mathbf{N}_s$
Data: $P = \{\mathbf{N}_0, \mathbf{E}_0, \mathbf{N}_1, \mathbf{E}_1, \ldots, \mathbf{E}_{K-1}, \mathbf{N}_K\}$ ▷ path, where $\mathbf{N}_0 = \mathbf{N}_s$ and $\mathbf{N}_K = \mathbf{N}_d$

1: $\mathbf{N}_s \leftarrow$ `Localise(` $\mathbf{M}$ `)` ▷ determine source node (current location)
2: $P \leftarrow$ `FindPath(` $\mathbf{N}_d$, $\mathbf{N}_s$ `)` ▷ find path to destination
3: $n \leftarrow 0$
4: $\mathbf{N}_\alpha \leftarrow P.node[n]$
5: **while** $\mathbf{N}_\alpha \neq \mathbf{N}_d$ **do** ▷ repeat until we are at the destination
6: $\mathbf{E}_\alpha \leftarrow P.node[n]$ ▷ retrieve next edge
7: $\mathbf{N}_\Omega \leftarrow P.node[n+1]$ ▷ and intermediate goal node
8: `TurnTo (` $\mathbf{E}_\alpha$`.bearing )`
9: $length \leftarrow \mathbf{E}_\alpha$`.length`
10: **repeat** ▷ repeat until we reach intermediate goal
11: **if** $\mathbf{E}_\alpha.type = real$ **then**
▷ find and follow road
12: $status \leftarrow$ `FollowRoad`$(length)$
13: **else**
▷ maintain bearing cross-country
14: $status \leftarrow$ `FollowBearing`$(length)$
15: **end**
16: **if** $status = blocked$ **then**
17: $\mathbf{E}_\alpha$`.status` $\leftarrow blocked$ ▷ note that this edge was blocked
18: `BacktrackTo (`$\mathbf{N}_\alpha$`)` ▷ return to previous node
19: $P \leftarrow$ `FindPath(` $\mathbf{N}_d$, $\mathbf{N}_\alpha$ `)` ▷ find new path from current node
20: **goto** 5
21: **else**
▷ determine how far we are from the intermediate goal
22: $length \leftarrow$ `DistanceBetween(Localise(` $\mathbf{M}$ `),` $\mathbf{N}_\Omega$`)`
23: **end**
24: **until** $length = 0$
25: $\mathbf{E}_\alpha$`.status` $= clear$ ▷ note that this edge was clear
26: $\mathbf{N}_\alpha \leftarrow \mathbf{N}_\Omega$
27: $n \overset{+}{\leftarrow} 1$
28: **end**

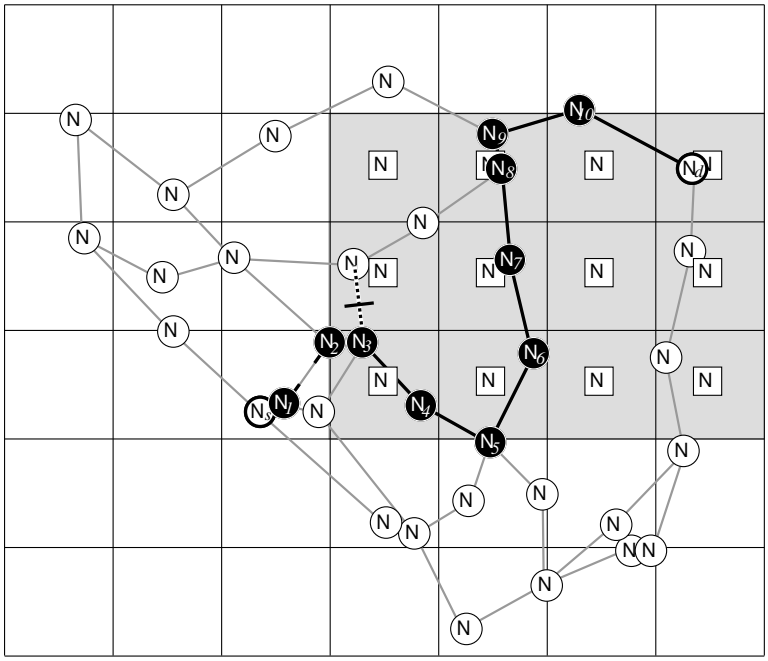


Figure 7.10: Blockage found, first re-plan

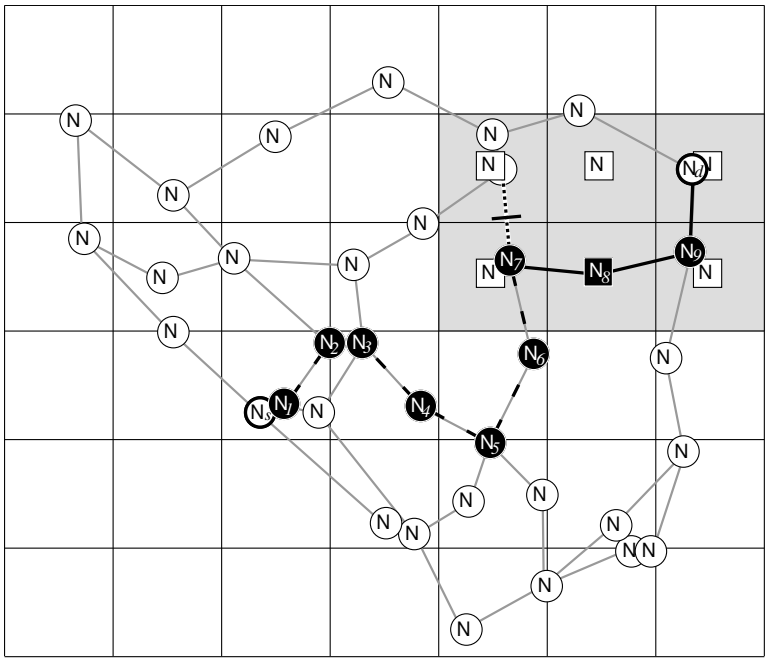


Figure 7.11: Second blockage found, second re-plan using pseudo-node

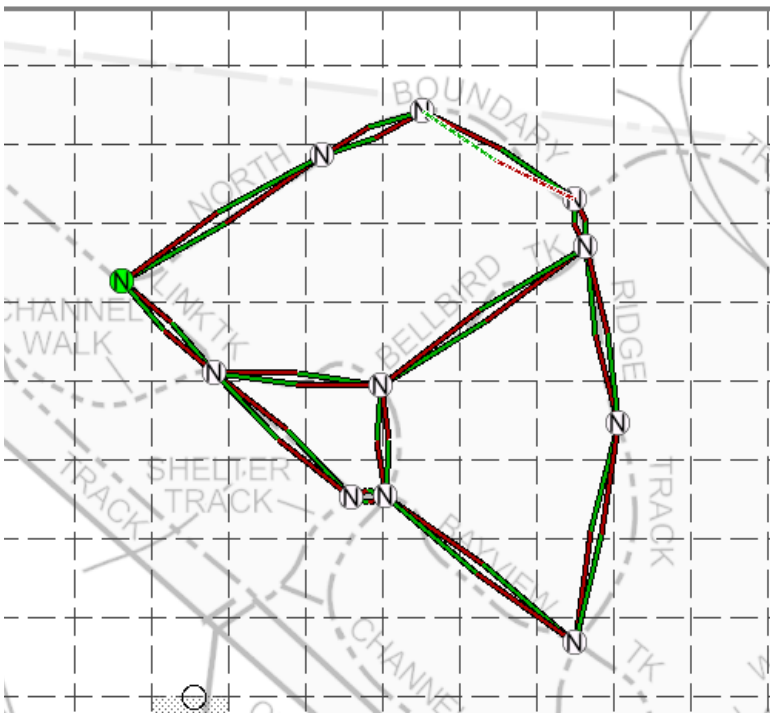


Figure 7.12: Example Map 1

Figure 7.13: Example Path 1

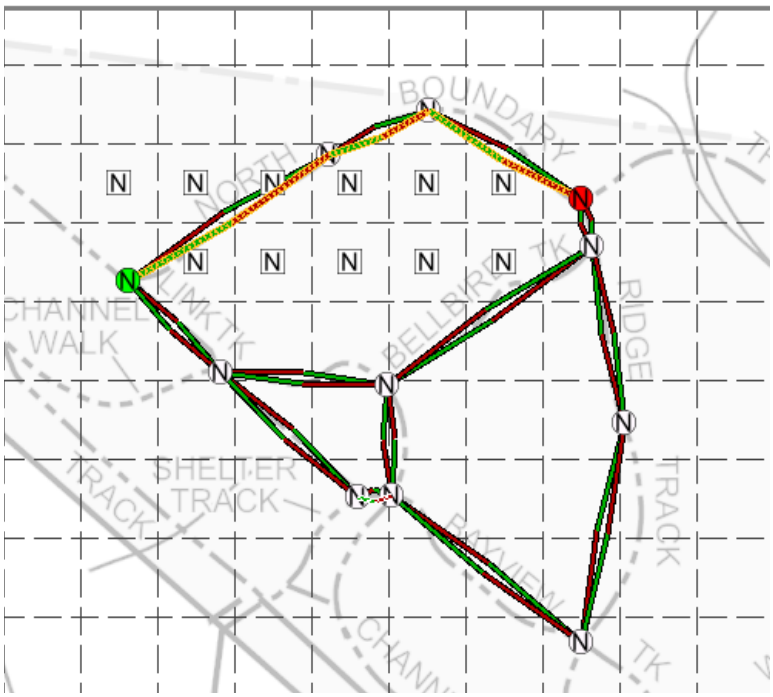


Figure 7.14: Example Path 2a: clear roads

is shown in yellow. To avoid clutter, pseudo-edges are not shown.

In Figure 7.14, the left-most node is the source and the top-right node is the destination. With no blockages and a reasonably direct path existing via roads, the resulting path is simple. Figure 7.15 has the same source and destination nodes, but now there is a known track blockage on the last leg, indicated by a red disc with a light cross. As such, a longer, but still track-based path is chosen. Figure 7.16 has a very similar scenario, but in this case the track blockage is unknown (a white disc with a dark cross) so, assuming clear tracks, the path chosen is the same as in Figure 7.14. Upon discovering the blockage, a backtrack is made to the last visited node and a new path is planned. This is shown in Figure 7.17 where the blockage is now known and the re-plan makes use of a single intermediate pseudo-node to make a cross-country detour.

Figures 7.18 and 7.15 show how a change in relative edge weights can affect the choice of path. The source and destination nodes are the same, but in Figure 7.18, pseudo-edges are given a weight of 1.5 while in Figure 7.19 the weight is 1.6. In both cases real edges have a weight of 1.0. The path chosen in Figure 7.18 includes a significant cross-country component and is quite direct. The path length is approximately 380 and the total path cost is 460. With an increase in the penalty of traversing cross-country, however slight, the path chosen in Figure 7.19 contains only track-based real edges but now deviates significantly from a straight-line path. The path length is now much greater at approximately 470 (an increase of approximately 23.5%), though the total path cost is only slightly higher at 470. With a cross-country penalty of 1.6, the original path now has a cost of 475, an increase of 25% over the same path with a cross-country weight of 1.5, but only slightly more expensive than

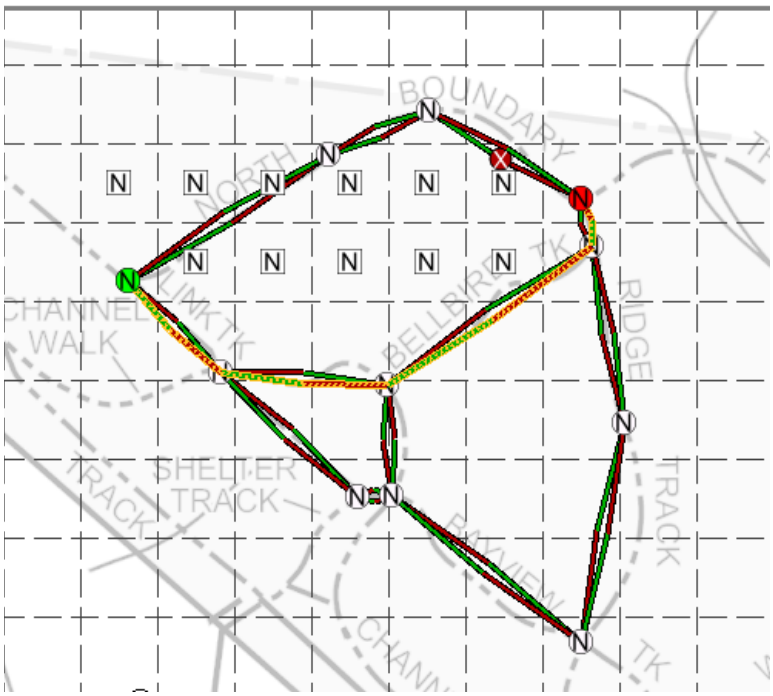


Figure 7.15: Example Path 2b: known blockage

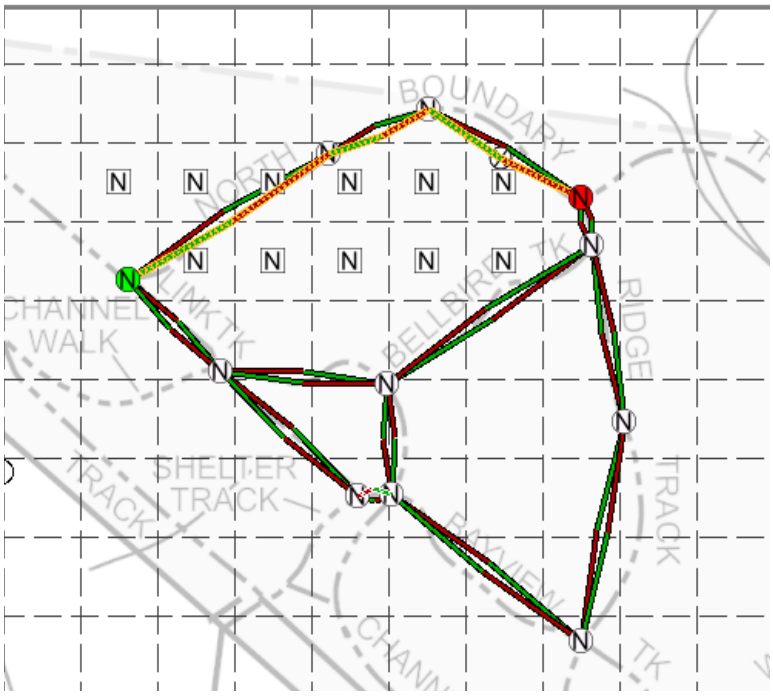


Figure 7.16: Example Path 2c: unknown blockage

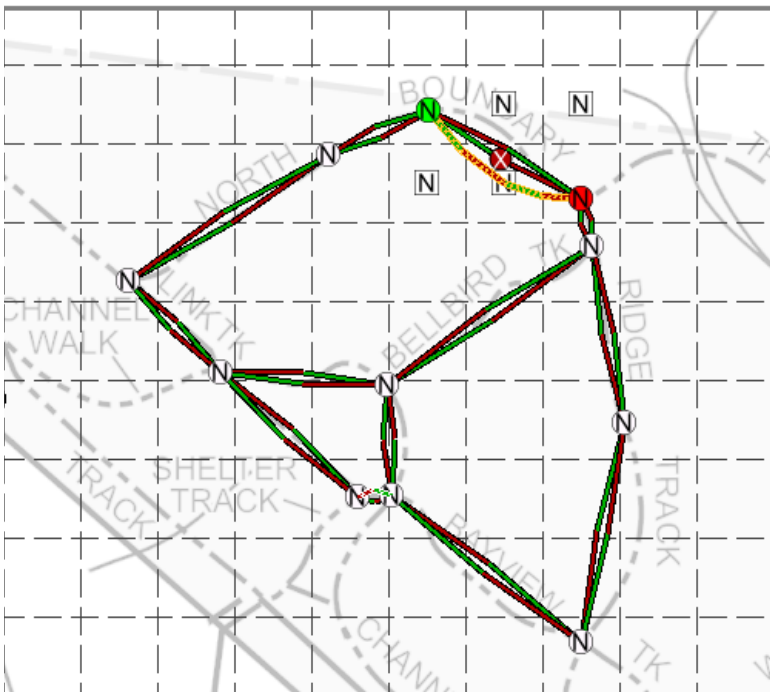


Figure 7.17: Example Path 2d: blockage discovered

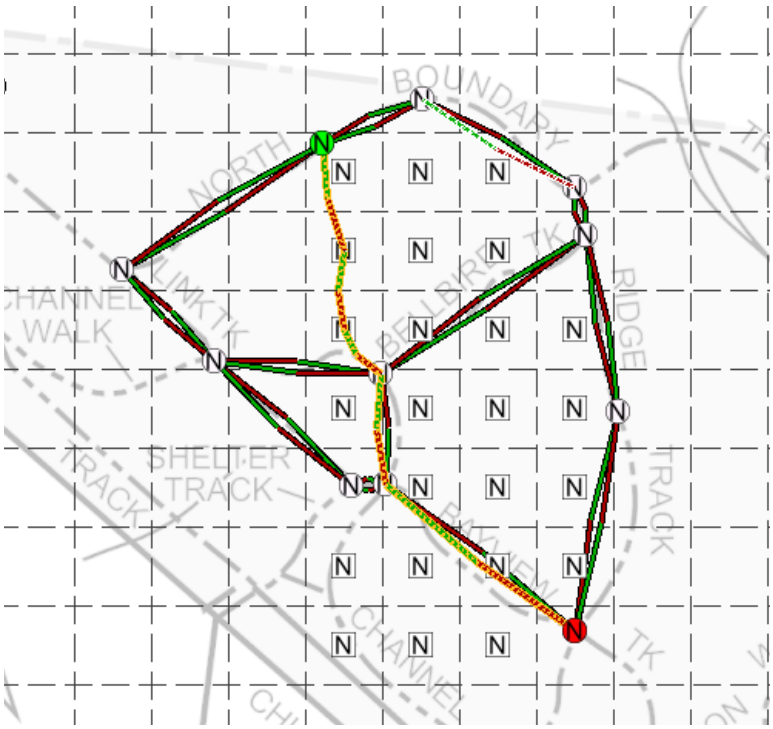


Figure 7.18: Example Path 3a: pseudo-edge weight of 1.5

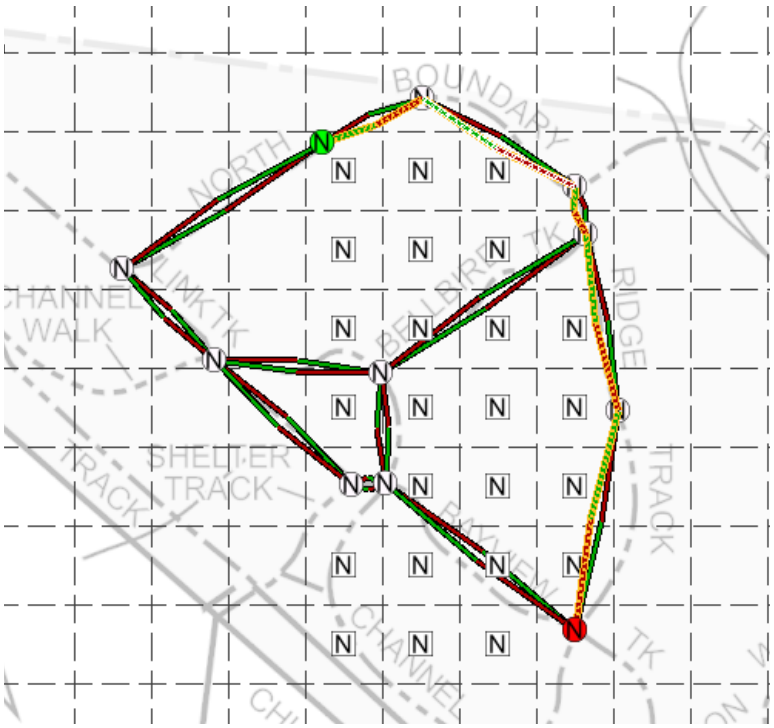


Figure 7.19: Example Path 3b: pseudo-edge weight of 1.6

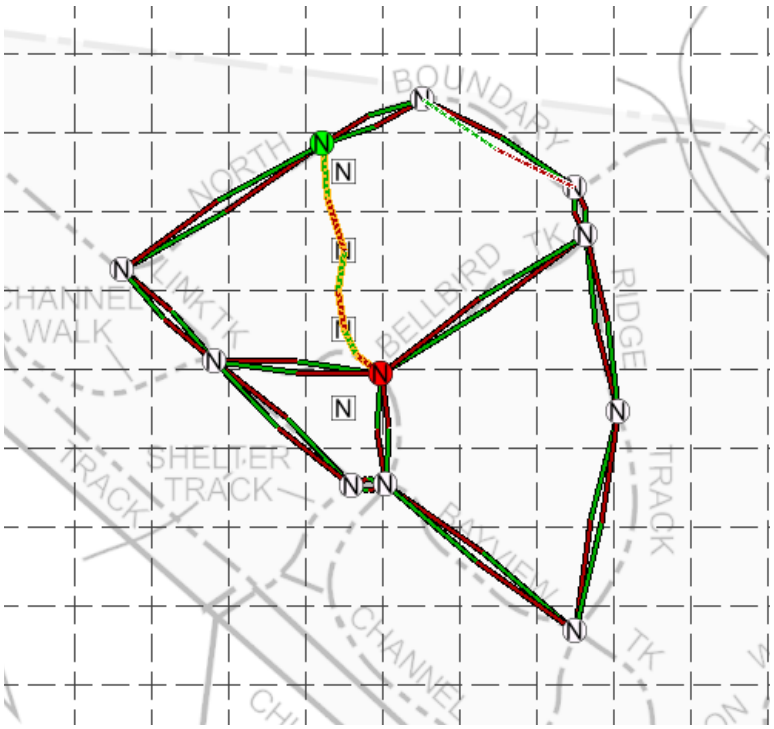


Figure 7.20: Example Path 4

the track-only path in 7.19. This clearly demonstrates the sensitivity of the path creation process to the relative edge weightings and in particular the dependence of the optimal path (optimal in terms of perceived cost) on the choice of weightings. Given that the focus is on safe and efficient traversal and *not* on the optimality of the chosen path, such sensitivity does not reduce the overall usability of the planner or the validity of the chosen paths.

A relatively high cross-country penalty of 2.0 will lead to paths that heavily favour tracks at the cost of total path lengths. It will still allow, however, the creation of cross-country paths when the savings are great enough, such as the situation shown in Figure 7.20.

7.10 Conclusion

This chapter presented a high-level path-planner aimed at finding efficient paths through bush environments that contain rough tracks. The planner uses manually created hybrid maps that combine metric "points-of-interest" – distinct geographic locations – with topological edges – the tracks themselves that describe the connectivity of points-of-interest. Points-of-interest – or nodes – are described primarily by a GPS coordinate. While this is a deviation from the theme of navigation using vision alone, it allows the dynamic creation of cross-country paths through terrain that is unknown and hence unable to be otherwise characterised. The use of GPS coordinates – which can be read off most if not all currently available maps – means no additional information is required *a priori*. The scope of research has not allowed for a practical implementation and integration of the system with the visual navigation systems so the ideas presented here have been implemented and demonstrated in simulation, results of which have been given.

Chapter 8

Conclusions and Future Work

> “ *I didn't jump. I took a tiny step, and there conclusions were.* ”
> **Buffy Summers, Buffy the Vampire Slayer, “Phases”**

> “ *The best way to predict the future is to invent it.* ”
> **Alan Kay, 1940–present**

In this final chapter we first present the conclusions drawn from the work, both with respect to each component separately and with regards to the work as a whole, reflecting upon the initial goals outlined in the opening chapters. We then conclude by discussing a number of modifications, enhancements and extensions that could be considered for future work in order to improve the current systems or to increase their abilities.

8.1 Conclusions

The body chapters within this thesis, each describing a piece of the overall navigation system, presented their own conclusions specific to each piece of work. Here we summarise the work and provide an overall conclusion, considering the goals and aims stated in the introductory material.

The thesis title, Visual Navigation on Rough Tracks in Bush Environments, embodies *what* is being done (robotic navigation), *where* it is being done (in bush environments and primarily on rough tracks) and *how* it is being done (using vision). The answer to the question *why* is fairly simple: robots are at their essence machines that are designed and created to assist in human tasks, and there are a number of human tasks – many, such as fire-fighting and search-and-rescue operations, performed in life-threatening or time-critical scenarios – that would greatly benefit from helper robots that could at least navigate themselves through these environments, if not perform additional duties.

As has been often reiterated, the target environment is essentially a natural, outdoor environment and as such is poorly structured, poorly maintained, irregular, inconsistent and at times highly unpredictable. There are many ways in which navigation in such conditions

is made very difficult, most if not all of which are the result of nature, and not man, being the controlling element. Poor-quality, degraded and irregular ground surfaces – whether on tracks or off – can make the task of simply driving a robot in a controllable and predictable manner a non-trivial task. While lacking the hustle and bustle of urban environments, the bush presents its own set of obstacles, including trees, fallen branches, potholes, ditches, embankments, exposed rocks and tree-roots, all of which must be detected and avoided to ensure safe travel. The structure that does exist in the form of rough tracks is typically poor at best. In place of asphalt and painted lines are a cobbled mix of dirt, leaf-litter, gravel and sand and messy indistinct edges where the surrounding foliage attempts to reclaim the land. Add to this the unpredictable and often harsh effects of sunlight and shadows, and the task of finding such tracks in order to perform efficient and effective navigation is made very difficult.

While very often overlooked, these sorts of environments have been tackled through earlier work, largely though the DARPA Grand Challenges. Invariably though, where previous attempts have presented essentially complete navigation systems, they have typically made use of elaborate sensor and processing suites, commonly coupled with corporate sponsorship to cover the high cost of such systems. While popular, passive vision is rarely if ever used alone. More often, cameras are combined with multiple LIDAR scanners and often RADAR. With the motivations of mechanical simplicity and low cost, coupled with a wealth of accumulated knowledge and the ability to use intuitive human insight, the work presented in this thesis instead makes use of passive vision for all local sensing. The only exception is the proposed use of GPS for use by a global path-planner.

This thesis presents five navigation components that combine to form a navigation system that is essentially complete. What is missing, due to time and equipment constraints and limitations, is a robust mechanical platform and associated drive control. However, given that the subsystems presented are independent of any particular platform and have only the mild requirements of a few cameras and a computer, it is believed that this is in no way detrimental to the demonstration and validation of the work.

The first component, Relative Localisation with Visual Odometry, presented in Chapter 3, provides a solution to the problem of relative localisation for the purpose of motion control. Specifically, it attempts to overcome the problems associated with traversal over rough, irregular and very often slippery ground surfaces by using vision and optical-flow techniques to observe and estimate the motion of the ground near the robot and hence estimate the robot's own motion. The greatest benefit of this solution is the independence of odometry from locomotion. While implemented on a traditional wheeled platform, the use of vision means it is applicable without modification to almost any land vehicle, whether it uses wheels, tracks or legs. Experimental results have shown the system to be accurate and robust enough for the purpose of medium-scale relative localisation, with positional errors in the order of five percent after fifty metres of travel achievable and able to be exceeded.

The second element, Obstacle Detection with Passive Stereo Vision, presented in Chapter 4, provides a means of detecting commonly encountered obstacles. The use of passive stereo vision allows a dense depth map to be created simply and cheaply when compared to the only other practical alternative of a movable planar LIDAR scanner, whose use – specifically when combined with a pan-tilt unit to give more than a single planar scan – requires careful calibration and control to avoid errors such as those caused by mechanical vibration and poor synchronisation of scanner movement and sensing. Using the obtained depth map and assuming the ground is locally flat, a local ground-plane estimate is formed against which

depth points can be compared. An obstacle likelihood map is then created where each point is assigned a likelihood of being an obstacle based on its absolute distance from the estimated ground-plane. In this manner, both positive obstacles – those that protrude above the ground, such as trees, rocks and fallen branches – and negative obstacles – those that lie below the ground, such as potholes and ditches – are equally able to be perceived. By using a locally determined ground estimate, as opposed to a predetermined one, we are able to accommodate local variations that would not impede travel but could easily be misclassified. Experimental results show that the combination of ground-plane estimation and point deviation is able to reliably detect both positive and negative obstacles including changes in terrain – such as a forked path that slopes down – that could be potentially hazardous.

The third piece, Detecting Poorly Structured Dirt Roads using Colour Vision, presented in Chapter 5, gives a solution to the problem of how poorly structured bush tracks – also referred to as simply dirt roads – may be found in order to allow the robot to be guided along them. In spite of their poor structure and the potential difficulty in finding and hence following them, dirt roads are of great navigational use because the represent regions of space that, by construction, offer a much greater likelihood of safe and efficient traversal compared to cross-country trekking. Investigations have lead to the observation that one of the most consistent distinguishing feature of these tracks is their visual appearance, specifically that their appearance is different to that of the surrounds, primarily because of a difference in material. Using passive colour vision, the road is able to be characterised following a statistical analysis of an image region that is assumed to contain predominately road. Each such descriptor creates a colour filter and a set of filters is accumulated over time, allowing the system to adapt to changes in road surface while maintaining a "*memory*" of previously observed road. This allows the system to cope comfortably with large variations in road appearance, most notably changes due to bright sunlight and strong shadows. Additionally, new filters are only created if they are likely to provide new information, that is, if they describe a region of colour-space not already described by an existing filter. Experimental results have shown the system capable of providing a distinct and accurate map of the road surface in spite of low contrast, multiple road materials, very poorly defined edges and the presence of bright sunlight and strong shadows simultaneously.

The fourth component, Local Path-Planning using Likelihood Maps, presented in Chapter 6, combines the results of the previous two sensing mechanisms to produce an overall *amenability map*. This describes the likelihood of a region being amenable to safe traversal, based on the criteria of safe space being free of obstacles and preferably belonging to the road. Following some simply geometric assumptions – primarily that the path the robot should take (typically along the road) flows generally away from the robot in the direction it is facing – the map is analysed and a local path is created that balances properties including directional smoothness, preferred bearing, overall amenability and path length. Experimental results show the system's ability to combine road and obstacle likelihood maps into amenability maps that accurately describe regions of safety. They also show the ability to plan a local path from this map that can guide the robot around obstacles, staying on the road when possible but also allowing traversal off-road when necessary to avoid an obstacle.

The final element, Global Path-Planning with a Hybrid Map, presented in Chapter 7 completes the overall navigation package by providing a global path planner. Using an existing road map of an area – and such maps are commonly available even for bush environments – a concise hybrid map is created that combines metric nodes – the location of distinct places of interest, such as intersections and waypoints along tracks – with topological edges

– a combination of actual tracks that connect nodes and potential straight-line cross-country paths. This allows large-scale paths to be planned that exploit tracks where possible but also allows for cross-country traversal if deemed beneficial. Results of simulations demonstrate how cross-country paths can be dynamically generated and used if they offer benefits.

In summary, the work presented in this thesis describes a set of navigation components that, while not perfect, is able to cope with a range of common detrimental environmental conditions, such as slippery terrain, poor structure and irregular lighting, and is believed to be an improvement on comparable existing systems. Of particular importance is the sole use of passive vision for the significant majority of the work, the one exception being the proposed use of GPS for localisation within the global path-planner. While similar tasks have been attempted before – most significantly through the DARPA Grand Challenges – few if any solutions have limited sensing to passive vision. This is seen as particularly important considering the relative costs of vision versus sensing alternatives and the financial resources typically available to potential users of such systems, especially the volunteer-based bush firefighters and rural search-and-rescue operations.

8.2 Future Work

In this section we discuss a number of avenues in which the work presented could be modified, enhanced or extended in the future. Some of the ideas presented are the result of recognising shortcomings of the current work, while others are ideas that could prove to be fruitful but were unable to be implemented and tested within the scope of the research.

8.2.1 Relative Localisation with Visual Odometry

Persistent Features: One of the benefits of the presented visual odometry system is its use of external features, that is, it is exteroceptive. As a result, it should be able to reduce drift errors through repeated use of features, that is, by making features persistent across more than a single pair of frames. This is not fully taken advantage of in the current system, however, primarily because of the associated additional processing and hence reduction in speed. Reliability would likely be increased, though, by referencing features more than once. For example, for a feature defined in frame N, instead of simply searching in frame $N-1$, the same feature could be sought in frame $N-2$ as well, with an appropriate change to the search window, such as using the feature displacement obtained from frame $N-1$ to determine the most likely location of the feature in frame $N-2$. Conflicting results could be resolved by a further match with frame $N-3$. Clearly, however, this would require many more feature-matching operations and as such would reduce speed significantly. This would need to be offset, such as by using fewer features per frame or utilising a better (that is, faster) feature-matching algorithm.

Adaptive Search Windows: In the current system, a fixed search window is used for matching features. The window is a rectangle of fixed dimensions and is centred on the feature itself. A more intelligent approach, one that was able to adapt to current motion, would be to use adaptive search windows. One such implementation would initially use a small number of key features (say four in each image) with large search windows. Each of these key features would then define an image region (for example, simply partitioning the image into equal

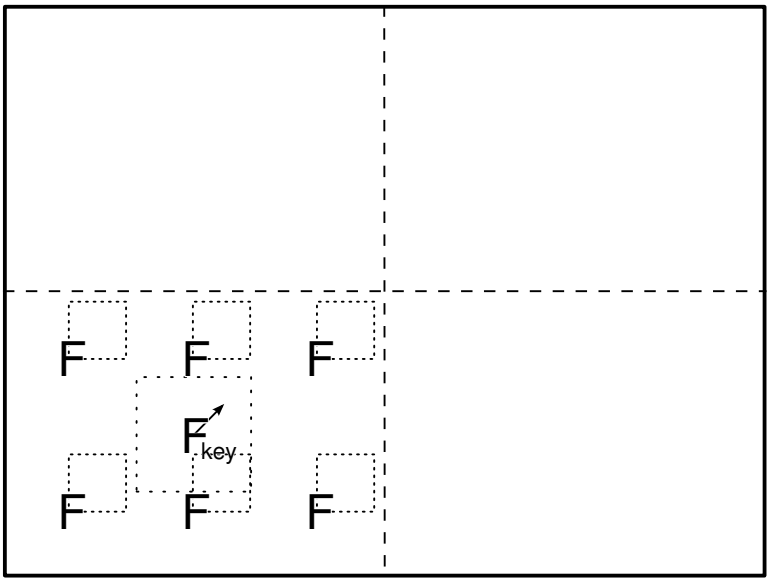


Figure 8.1: Adaptive search windowing

quadrants). Then, for each of the remaining much larger number of features, each feature's search window would be much smaller and centred on a point corresponding to an assumed displacement the same as that recovered from the key feature. This is shown diagrammatically in Figure 8.1. The displacement of the key feature $\mathbf{F}_{key}$ in the centre of the lower-left quadrant is found first, using a large search window centred on the feature. The search windows of the remaining features in that quadrant are smaller but are shifted by the estimated displacement of $\mathbf{F}_{key}$. This extension aims to reduce the search space and hence reduce computation and improve performance. While the initial search windows are larger than would otherwise be used (and hence require greater processing), subsequent windows can be made smaller, meaning the overall number of feature comparisons is reduced. Additionally, this could make the required processing effectively independent of the maximum robot speed to be estimated. In the current system, without knowing where to look, the only way to increase the maximum estimated feature displacement (and hence the maximum robot speed for a given frame-rate) is to increase the size of the search window. This, however, increases the processing required, therefore potentially reducing the frame-rate and reducing the measurable robot speed. We aim here to exploit the inherent single-rigid-body assumption, that is, the assumption that we are estimating the motion of a single, rigid body, that being the ground. Under this assumption, the displacement of points on the body are locally similar (though in the case of rotations, not identical). As such, we assume that the displacement of one point (each of the key features) is representative of the displacement of surrounding points, and use this to modify the feature search windows. Measuring a higher robot speed means a larger search window for a small number of (key) features but for the majority of features this now means simply increasing the displacement of the search window.

Dedicated/Parallel processing: A great deal of the processing time taken to estimate motion is spent matching features, specifically calculating the similarity of a feature with a

Figure 8.2: The Point Grey Research® Inc. BumbleBee 2® stereo system [110]

potential match. In the current implementation – utilising the sum of absolute differences – this is a very simple operation in terms of computational complexity and is therefore a prime candidate for implementation in either hardware or firmware. This would take the form of an off-board, dedicated processing unit, for example using a Field Programmable Gate Array device or a microcontroller. In addition, because the process of matching a feature is independent of other features and with such hardware being relatively cheap, it is conceivable – and would be highly advantageous – to use multiple such units and process in parallel. Each such unit could be given a feature (an image patch) and a search region (a larger image patch) and simply return the displacement of the best match. A further extension to this would be to implement the entire visual odometry system in dedicated hardware, allowing the system to simply be plugged in to almost any platform and provide odometry.

8.2.2 Obstacle Detection with Passive Stereo Vision

General Improvements: The single biggest problem with the stereo system in its current state is speed. With each image pair taking in the order of a few seconds to process – with the significant majority of this time being taken finding correspondences – the detection of obstacles, and hence the planning of local paths, cannot be done rapidly enough for the system to be effectively reactive. It can therefore not respond appropriately to dynamic obstacles. However, with stereo vision being simply a sensory tool within the context of this work, it is sufficient to simply note that hardware solutions exist, such as the *BumbleBee 2* from Point Grey Research [109], shown in Figure 8.2, that can produce dense disparity images at high a framerate and at a high resolution (stated specifications are 640×480 at 48 frames-per-second or 1024×768 at 18 frames-per-second), though most if not all current off-the-shelf hardware solutions are of fixed and relatively small baseline (the BumbleBee 2 for instance comes with a fixed baseline of either 12 or 24 centimetres), which limits their use for navigation in outdoor environments.

More relevant to the work presented, there are ways in which the obstacle detection system could be improved, primarily with regards to the modelling and estimation of the ground:

Ground height constraints: The current system assumes a planar local terrain whose normal does not vary significantly from that of a calibrated ground plane. However, there are no constraints on the distance between the estimated ground and the calibrated ground meaning, were the robot faced with a step change in the terrain where the transition occurred within the sensor blind-spot, the system could incorrectly consider the ground beyond the step

to be safe, ignoring the potentially dangerous rise or drop. Presumably, though, the change in ground level would have been sensed before it entered the blind-spot and hence avoided, though tighter constraints that consider both the plane normal *and* its position would be an improvement.

Curved ground model: In addition, the use of a plane is potentially over-restrictive, especially in outdoor natural terrain. Gentle undulations that would present no impediment to traversal are not comfortably accommodated. Accordingly, the use of a more general ground model would allow such curvature to be taken into account. For example, modelling the ground as part of a (large) sphere and placing a lower limit on the sphere radius would allow curvature below a certain level.

8.2.3 Detecting Poorly Structured Dirt Roads using Colour Vision

Perhaps the largest potential for erroneous behaviour within the road-finding system comes from the assumption that the road-characterising window always contains a region of road. If this region ever contains non-road areas for long enough for all memory of the true road to be lost, a form of positive feedback will occur and the robot will remain off the road, falsely believing the non-road areas to be where it should stay. There are two ways in which this could potentially be alleviated, both centred on more selective characterisation.

Switched characterisation: The first involves a simple switching of the characterisation process. In the cases when the robot is required to travel off the road, a common example being when it needs to avoid a large obstacle on the road, the characterisation process should temporarily be suspended so that new filters are not made and only existing road filters are used. In this manner, earlier correct characterisations of the road will be retained meaning only the true road will continue to be extracted, whether it is present in the defining region or not. When the robot returns to the road, characterisation can resume, again allowing it to adapt to local changes.

Terrain-selective characterisation: The second improvement involves a tighter coupling of the road-finder with the obstacle-finder. Having determined safe terrain in terms of the location of obstacles, the obstacle map can be used as a characterisation mask. Within the predetermined road window, only areas that are known to be safe (that is, are not part of an obstacle) should be used to characterise the road. This further emphasises the notion that we wish the robot to travel along the road but *only* where there are no obstacles, by only using obstacle-free areas to characterise the road.

8.2.4 Local Path-Planning using Likelihood Maps

Relaxed geometry: A potential improvement would be the use of a robo-centric polar amenability map instead of the current Cartesian map. More specifically, the use of annuli instead of simple rectangular slices to separate the map into regions would relax the assumption that the desired path flows in the direction the robot is facing. The (valid) assumption that the desired path begins at the robot and moves away would still hold, but annuli would better accommodate situations where the robot is approaching the road or its desired bearing

from an angle, as would be the case when the robot rejoins the road after avoiding an obstacle or performs cross-country traversal.

8.2.5 Global Path-Planning with a Hybrid Map

There are a number of ways in which the global path-planner could be improved, though a practical implementation and experimentation would no doubt reveal many more subtle avenues for improvement.

Temporally-variable edge status: Currently, once a map edge has been flagged as being blocked, that edge will no longer be considered for use. In a long-term practical sense this is extremely limiting, as, without manual intervention, the robot will have no opportunity to investigate and determine whether or not that edge ever becomes usable again. A situation could easily be imagined where, over a long enough period, more and more edges are found to be blocked and the map contains fewer and fewer usable edges, in spite of the fact that we could reasonably assume that any blockages would be cleared within a relatively short period of their discovery. A solution would be to time-decay the blocked status. A simple version would see a status of *blocked* being changed to *unknown* after a fixed period of time. After that period the edge would again be considered for use. If the physical blockage remained, the edge status would again be changed to *blocked* and the process would repeat. If the physical blockage had been cleared, the edge status would be set to *clear* and the edge would continue to be available for consideration. This is an embodiment of the simple assumption that any blockage, once discovered, is likely to have been cleared after a period of time. A slightly more advanced version would have the edge status take a numerical value that corresponds to the likelihood of the edge being clear. For example, a clear or unknown (but assumed clear) edge has a value of 1.0 and an edge known to be blocked has a value of 0.0. This value can then be used as an edge weight modifier, for example, as an edge weight divisor. Then, over time, the status value can be gradually increased to a maximum of 1.0 . This corresponds to the assumption that, the more time that has passed since an edge was discovered to be blocked, the more likely it is that the blockage has been cleared. Either of these solutions would allow the robot to adapt to the long-term changes in the environment.

Dynamic edge weights: In a similar vein, edge cost-per-distance weights are currently temporally fixed and indeed are constant, with real edges having one cost and pseudo-edges having another, higher, cost. A better idea would be to have variable costs that are determined by the robot *in situ*. With all unseen edges given the same initial cost, the robot could adjust the cost of an edge after it has been traversed to reflect the ease with which the edge was traversed. For example, given the (approximately) known edge length and an assumed average robot speed, a comparison could be made between a predicted traversal time and the actual traversal time. Edges that can be traversed in a shorter time than predicted would have their cost reduced while edges that take longer to traverse than predicted would have their cost increased. In this manner, the map would reflect a great many environmental features, such as the specific nature of the ground surface (rough versus smooth, slippery versus high-traction) and changes in elevation (flat versus uphill versus downhill) that are both difficult and unnecessary to cater for individually.

8.2.6 General Extensions

In addition to the improvements and extensions specific to the current realisation of the proposed navigation system, there are a number of areas in which a robotic system as a whole, equipped with such a navigator, could be extended.

The robotic platform itself could be improved in many ways. With the use of visual odometry that is independent of the form of locomotion, this system could be used on a robot with tracks or – perhaps in addition to wheels – legs that would mean many track blockages, such as fallen trees, could be traversed instead of causing a retreat or re-plan.

For a more task-specific advancement, a search-and-rescue robot would benefit greatly from the addition of further sensors, such as thermal cameras and microphones, that could assist in finding people. In these sorts of missions, where covertness is not required, equipping the robot with floodlights or spotlights to allow use even in otherwise completely dark situations would be viable and advantageous. This scenario would also greatly benefit from cooperative robotics, a good example being the use of both ground-based robots to search tracks and aerial robots to help guide and coordinate. At a level of even greater activity, a robot with arms that could collect and safely carry an injured person – perhaps even supplying basic medical assistance through the monitoring of heart rate and blood pressure – would be invaluable.

For firefighting robots, the most obvious addition would be some form of firefighting equipment that could be used to extinguish spot-fires as they are found and before they get too large. Additional sensors, measuring properties such as air temperature, humidity and wind speed and direction could assist firefighters by giving them notice of changes. Such information could even be used by the robot itself to determine the likely source of a newly-detected fire which it could then proceed to find and extinguish. In a smokey environment where visibility is often poor, additional sensors, such as a LIDAR rangefinder, would provide a backup to passive sensors.

Park maintenance robots would be made far more useful if they could not only navigate and detect blocked tracks, but if they could remove the blockage or at least allow a human operator to do so remotely. This could take the form of relatively simple cutting equipment coupled with forklift arms that could firstly cut fallen trees into manageable pieces and then lift and move the pieces out of the way. Alternatively, a bulldozer-like actuator could be used to re-form parts of washed out tracks or again to push trees and other similar debris out of the way.

Bibliography

[1] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, pp. 224–241, 1992. [Online]. Available: http://ieeexplore.ieee.org/iel4/21/3929/00148426.pdf

[2] B. Bartlett, V., Estivill-Castro, and S. Seymon, "Dogs or robots: why do children see them as robotic pets rather than canine machines?" in *Fifth Conference on Australasian User Interface*, 2004, pp. 7–14. [Online]. Available: http://crpit.com/confpapers/CRPITV28Bartlett.pdf

[3] P. H. Batavia and S. Singh, "Obstacle detection using adaptive color segmentation and color stereo homography," in *IEEE International Conference on Robotics and Automation*, 2001, pp. 705–710. [Online]. Available: http://ieeexplore.ieee.org/iel5/7423/20179/00932633.pdf

[4] M. Bertozzi, A. Broggi, and A. Fascioli, "Vision-based intelligent vehicles: State of the art and perspectives," *Robotics and Autonomous Systems*, vol. 32, no. 1, pp. 1–16, Jul 2000. [Online]. Available: http://www.sciencedirect.com/science/article/B6V16-40D0NDV-1/2/cf4ff131e3a0e11a15d4182766dab5c8

[5] M. Bertozzi, A. Broggi, M. Cellario, A. Fascioli, P. Lombardi, and M. Porta, "Artificial vision in road vehicles," *Proceedings of the IEEE*, vol. 90, pp. 1258–1271, Jul 2002. [Online]. Available: http://ieeexplore.ieee.org/iel5/5/22179/01032807.pdf

[6] J. Blitch, "Artificial intelligence technologies for robot assisted urban search and rescue," *Expert Systems with Applications*, vol. 11, no. 2, pp. 109–124, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/B6V03-3WP2C28-3/1/e6a587c171d78b99820f31e9f86b2d8a

[7] J. Borenstein and L. Feng, "Gyrodometry: a new method for combining data from gyros and odometry in mobile robots," in *IEEE International Conference on Robotics and Automation*, vol. 1, Apr 1996, pp. 22–28. [Online]. Available: http://ieeexplore.ieee.org/iel3/3678/10815/00503813.pdf

[8] J. Borenstein, H. Everett, and L. Feng, *Navigating Mobile Robots: Sensors and Techniques*. A. K. Peters, Ltd., Wellesley, MA, 1996. [Online]. Available: http://www-personal.engin.umich.edu/~johannb/my_book.htm

[9] T. Bosch and M. Lescure, Eds., *Selected Papers on Laser Distance Measurements*. SPIE Optical Engineering Press, 1995.

[10] J.-Y. Bouguet, "Camera calibration toolbox for matlab," 2006, visited 2006. [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html

[11] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, pp. 14–23, Mar 1986. [Online]. Available: http://ieeexplore.ieee.org/iel5/8149/23631/01087032.pdf

[12] R. Bunschoten and B. Kröse, "Visual odometry from an omnidirectional vision system," in *International Conference on Robotics andAutomation*, vol. 1, September 2003, pp. 577–583. [Online]. Available: URL:http://ieeexplore.ieee.org/iel5/8794/27829/01241656.pdf

[13] Carnegie Mellon Red Team Robot Racing, "Carnegie mellon red team robot racing," 2004, visited 04/2005. [Online]. Available: http://www.redteamracing.org

[14] H. E. A. S. A. R. Center, "The apollo 15 lunar rover on the moon," 2007, visited 2007. [Online]. Available: http://starchild.gsfc.nasa.gov/docs/StarChild/space_level2/apollo15_rover.html

[15] H. T. Chan, K. S. Tam, and N. K. Leung, "A neural network approach for solving the path planning problem," in *IEEE International Symposium on Circuits and Systems*, 1993. [Online]. Available: http://ieeexplore.ieee.org/iel2/1067/8956/00394261.pdf

[16] R. Chatila and J. Laumond, "Position referencing and consistent world modeling for mobile robots," in *IEEE International Conference on Robotics and Automation*, 1985, pp. 138?–143. [Online]. Available: http://ieeexplore.ieee.org/iel5/8151/23642/01087373.pdf

[17] D. Ciccimaro, H. Everett, H. Gilbreath, and T. Tran, "An automated security response robot," in *Mobile Robots XIII and Intelligent Transportation Systems*, 1998, pp. 50–61.

[18] D. Ciccimaro, H. Everett, M. Bruch, and C. Phillips, "A supervised autonomous security response robot," in *American Nuclear Society 8th International Topical Meeting on Robotics and Remote Systems*, Apr 1999. [Online]. Available: http://citeseer.nj.nec.com/ciccimaro99supervised.html

[19] CMU Vision & Autonomous Systems Center, "Vision and autonomous systems center's image database," 2003, visited 2006. [Online]. Available: http://vasc.ri.cmu.edu//idb/

[20] P. Corke, D. Symeonidis, and K. Usher, "Tracking road edges in the panospheric image plane," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, Oct 2003, pp. 1330–1335. [Online]. Available: http://ieeexplore.ieee.org/iel5/8832/27959/01248829.pdf

[21] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*. MIT Press, 2001.

[22] J. Crisman and C. Thorpe, "Unscarf-a color vision system for the detection of unstructured roads," in *IEEE International Conference on Robotics and Automation*, vol. 3, Apr 1991, pp. 2496–2501. [Online]. Available: http://ieeexplore.ieee.org/iel2/347/3640/00132000.pdf

[23] DARPA, "Darpa grand challenge 2004," 2004, visited 03/2005. [Online]. Available: http://www.darpa.mil/grandchallenge04/

[24] ——, "Darpa grand challenge 2005," 2005, visited 03/2005. [Online]. Available: http://www.darpa.mil/grandchallenge05/

[25] ——, "Darpa grand challenge," 2006, visited 03/2005. [Online]. Available: http://www.darpa.mil/grandchallenge/

[26] K. Dautenhahn, S. Woods, C. Kaouri, M. L. Walters, K. L. Koay, and I. Werry, "What is a robot companion - friend, assistant or butler?" in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 1192–1197. [Online]. Available: http://ieeexplore.ieee.org/iel5/10375/32977/01545189.pdf

[27] DICKEY-John Corporation, "Radar iii," 2007, visited 2007. [Online]. Available: http://www.dickey-john.com/products/ag/product_detail.php?products_id=98

[28] A. Doerr, "The anatomy of an intelligent robot," in *2nd European Conference on Automated Manufacturing*, 1983, pp. 393–396.

[29] J. Dolan, A. Trebi-Ollennu, A. Soto, and P. Khosla, "Distributed tactical surveillance with atvs," in *SPIE Proceedings on Unmanned Ground Vehicle Technology*, 1999, pp. 192–199.

[30] A. Elfes and L. Matthies, "Sensor integration for robot navigation: combining sonar and range date in a grid-based representation," in *26th IEEE Conference on Decision and Control*, vol. 3, Dec 1987, pp. 1802–1807.

[31] I. Erkmen, A. Erkmen, F. Matsuno, R. Chatterjee, and T. Kamegawa, "Snake robots to the rescue!" *IEEE Robotics & Automation Magazine*, vol. 9, no. 3, pp. 17–25, Sep 2002. [Online]. Available: http://ieeexplore.ieee.org/iel5/100/22218/01035210.pdf

[32] H. Everett, *Sensors for Mobile Robots: Theory and Application*. A. K. Peters, Ltd, 1995.

[33] O. Faugeras, B. Hotz, H. Mathieu, T. Viéville, Z. Zhang, P. Fua, E. Théron, L. Moll, G. Berry, J. Vuilleminand, P. Bertin, and C. Proy, "Real time correlation based stereo: algorithm implementations and applications," INRIA, Tech. Rep., 1993. [Online]. Available: http://perception.inrialpes.fr/Publications/1993/FHMVZFTMBVBP93

[34] Feathertop Mapping Services, "Feathertop and rooftop maps," http://www.feathertopmapping.biz/, 2006, visited 2006. [Online]. Available: http://www.feathertopmapping.biz

[35] Federal Aviation Administration, "Specification for the wide area augmentation system," 2001, visited 2007. [Online]. Available: http://gps.faa.gov/Library/Data/waas/2892bC2a.pdf

[36] J. Fernández and A. Casals, "Autonomous navigation in ill-structured outdoor environment," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 1997, pp. 395–400. [Online]. Available: http://ieeexplore.ieee.org/iel4/5229/14134/00649093.pdf

[37] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981. [Online]. Available: http://portal.acm.org/citation.cfm?id=358692&dl=ACM&coll=portal

[38] J. Folkesson and H. Christensen, "Graphical slam - a self-correcting map," in *IEEE International Conference on Robotics and Automation*, 2004. [Online]. Available: http://ieeexplore.ieee.org/iel5/9126/29020/01307180.pdf

[39] Y. Fujita, Y. Nakamura, and Z. Shiller, "Dual dijkstra search for paths with different topologies," in *IEEE International Conference on Robotics and Automation*, 2003. [Online]. Available: http://ieeexplore.ieee.org/iel5/8794/27835/01242109.pdf

[40] Fujitsu, "Fujitsu robot project: Hoap series," 2007, visited 2007. [Online]. Available: http://www.fujitsu.com/global/about/rd/200506hoap-series.html

[41] J. Fulton and J. Pransky, "Darpa grand challenge - a pioneering event for autonomous robotic ground vehicles," *Industrial Robot*, vol. 31, no. 5, pp. 414–422, 2004.

[42] S. Ge and Y. J. Cui, "New potential functions for mobile robot path planning," *IEEE Transactions on Robotics and Automation*, vol. 16, pp. 615–620, 2000. [Online]. Available: http://ieeexplore.ieee.org/iel5/70/19052/00880813.pdf

[43] R. Ghurchian, T. Takahashi, Z. Wang, and E. Nakano, "On robot self-navigation in outdoor environments by color image processing," in *7th International Conference on Control, Automation, Robotics and Vision*, vol. 2, Dec 2002, pp. 625–630. [Online]. Available: http://ieeexplore.ieee.org/iel5/8741/27724/01238496.pdf

[44] A. Giachetti, M. Campani, and V. Torre, "The use of optical flow for road navigation," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 34–48, Feb 1998. [Online]. Available: http://ieeexplore.ieee.org/iel4/70/14372/00660838.pdf

[45] GNU, "Gnu scientific library," 2006, visited 2006. [Online]. Available: http://www.gnu.org/software/gsl/

[46] Google, "Google maps australia beta," 2007, visited 2007. [Online]. Available: http://maps.google.com.au/maps?f=q&hl=en&q=churchill+park,+victoria&sll=-25.335448,135.745076&sspn=39.716136,74.003906&layer=&ie=UTF8&om=1&z=15&ll=-37.950729,145.256038&spn=0.017191,0.048494&t=h

[47] ——, "Google maps," 2007, visited 2007. [Online]. Available: http://maps.google.com

[48] ——, "Google earth," 2007, visited 2007. [Online]. Available: http://earth.google.com

[49] A. Graves and C. Czarnecki, "A man-machine interface for an ai-based telerobotic system," in *International Conference on Human Interfaces in Control Rooms, Cockpits and Command Centres*, 1999, pp. 302–307. [Online]. Available: http://ieeexplore.ieee.org/iel5/6382/17061/00787726.pdf

[50] X. Guangming, X. Zhengfei, G. Junyao, and H. Zhimin, "Overcoming obstacles of mobile robots based on teleoperation and locally autonomous control," *Computer Measurement & Control*, vol. 14, pp. 193–195, 2006.

[51] S. Harmon, "The ground surveillance robot (gsr): An autonomous vehicle designed to transit unknown terrain," *IEEE Journal of Robotics and Automation*, vol. 3, no. 3, pp. 266–279, Jun 1987. [Online]. Available: http://ieeexplore.ieee.org/iel5/8149/23637/01087091.pdf

[52] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, 1968. [Online]. Available: http://ieeexplore.ieee.org/iel5/4082035/4082123/04082128.pdf

[53] L. J. Heyer, S. Kruglyak, and S. Yooseph, "Exploring expression data: Identification and analysis of coexpressed genes," *Genome Research*, vol. 9, no. 11, pp. 1106–1115, Nov 1999. [Online]. Available: http://www.genome.org/cgi/content/full/9/11/1106

[54] H. Hirschmüller, "Improvements in real-time correlation-based stereo vision," in *IEEE Workshop on Stereo and Multi-Baseline Vision*, 2001. [Online]. Available: http://ieeexplore.ieee.org/iel5/7750/21297/00988772.pdf

[55] Honda Motor Co., "Asimo," 2007, visited 2007. [Online]. Available: http://world.honda.com/ASIMO/

[56] I. P. Howard and B. J. Rogers, *Binocular vision and stereopsis*. Oxford University Press, 1995.

[57] Y. Hu and S. X. Yang, "A knowledge based genetic algorithm for path planning of a mobile robot," in *IEEE International Conference on Robotics and Automation*, 2004. [Online]. Available: http://ieeexplore.ieee.org/iel5/9126/28923/01302402.pdf

[58] Intel Corporation, "Open source computer vision library," 2006, visited 2006. [Online]. Available: http://www.intel.com/technology/computing/opencv/index.htm

[59] International Organization for Standardization, "Iso standard 8373:1994, manipulating industrial robots – vocabulary," 1994.

[60] iRobot Corporation, "Roomba vacuuming robot," 2007, visited 2007. [Online]. Available: http://www.roomba.com.au/

[61] R. A. Jarvis, "A perspective on range finding techniques for computer vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5, no. 2, pp. 122–139, Mar 1984.

[62] ——, "A stereo vision teleoperated robotic vehicle," in *Second Australian and New Zealand Conference on Intelligent Information Systems*, 1994, pp. 145–149. [Online]. Available: http://ieeexplore.ieee.org/iel2/3167/8985/00396932.pdf

[63] ——, "Natural landmark based robot navigation in distance transform space," in *8th Australian Joint Convention on Artificial Intelligence*, 1995, pp. 323–330.

[64] R. A. Jarvis and J. C. Byrne, "Robot navigation: touching, seeing and knowing," in *Australian Conference on Artificial Intelligence*, 1986.

[65] D. Kang, J. Choi, and I. Kweon, "Finding and tracking road lanes using "line-snakes"," in *1996 IEEE Intelligent Vehicles Symposium*, Sep 1996, pp. 189–194. [Online]. Available: http://ieeexplore.ieee.org/iel3/4275/12309/00566376.pdf

[66] A. Kelly, "Contemporary feasibility of image mosaic based vehicle position estimation," in *International Conference on Robotics and Applications*, Oct 1999.

[67] A. Kemurdjian, V. Gromov, V. Mishkinyuk, V. Kucherenko, and P. Sologub, "Small marsokhod configuration," in *IEEE International Conference on Robotics and Automation*, vol. 1, 1992, pp. 165–168. [Online]. Available: http://ieeexplore.ieee.org/iel2/399/5764/00220318.pdf

[68] L. Kleeman, "Advanced sonar sensing," in *Tenth International Symposium on Robotics Research*, R. Jarvis and A. Zelinsky, Eds., vol. 6. Springer, 2003, pp. 485–498.

[69] ——, "Advanced sonar with velocity compensation," *International Journal Robotics Research*, vol. 23, no. 2, pp. 111–126, Feb 2004. [Online]. Available: http://www.ecse.monash.edu.au/centres/irrc/LKPubs/IJRR1484.pdf

[70] G. Kowadlo, "Robot odour localisation in enclosed and cluttered environments using naïve physics," Ph.D. dissertation, Intelligent Robotics Research Centre, Department of Electrical and Computer Systems Engineering, Monash University, Australia, 2006.

[71] D. Kuan and U. Sharma, "Model based geometric reasoning for autonomous road following," in *IEEE International Conference on Robotics and Automation*, vol. 4, Mar 1987, pp. 416– 423. [Online]. Available: http://ieeexplore.ieee.org/iel5/8153/23644/01088049.pdf

[72] B. Kuipers and Y. Byun, "A qualitative approach to robot exploration and map-learning," in *Workshop on Spatial Reasoning and Multi-Sensor Fusion*, Oct 1987, pp. 390–404.

[73] J.-C. Latombe, *Robot motion planning*, ser. Kluwer international series in engineering and computer science. Boston: Kluwer Academic Publishers, 1991.

[74] J. Leonard and H. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," in *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, vol. 3, 1991, pp. 1442–1447. [Online]. Available: http://ieeexplore.ieee.org/iel2/561/4454/00174711.pdf

[75] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, 1982. [Online]. Available: http://ieeexplore.ieee.org/iel5/18/22725/01056489.pdf

[76] A. Lorusso and E. D. Micheli, "An approach to obstacle detection and steering control from optical flow," in *IEEE Intelligent Vehicles Symposium*, Sep 1996, pp. 357–362. [Online]. Available: http://ieeexplore.ieee.org/iel3/4275/12309/00566406.pdf

[77] R. Lotufo, A. Morgan, E. Dagless, D. Milford, J. Morrissey, and B. Thomas, "Real-time road edge following for mobile robot navigation," *Electronics & Communication Engineering Journal*, vol. 2, no. 1, pp. 35–40, Feb 1990. [Online]. Available: http://ieeexplore.ieee.org/iel1/2219/2620/00080014.pdf

[78] Lunar and Planetary Department, Sternberg State Astronomical Inst., Moscow University, "Lunokhod 1," 2006, visited 2007. [Online]. Available: http://selena.sai.msu.ru/Home/Spacecrafts/Lunokhod1/lunokhod1e.htm

[79] Y. Mae, A. Yoshida, T. Arai, K. Inoue, K. Miyawaki, and H. Adachi, "Application of locomotive robot to rescue tasks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000, pp. 2083–2088. [Online]. Available: http://ieeexplore.ieee.org/iel5/7177/19356/00895278.pdf

[80] M. Marzouqi and R. A. Jarvis, "Covert path planning in unknown environments with known or suspected sentry location," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005. [Online]. Available: http://ieeexplore.ieee.org/iel5/10375/32977/01545483.pdf

[81] M. S. Marzouqi and R. A. Jarvis, "Distance transform based gaussian distribution for probabilistic target tracking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006. [Online]. Available: http://ieeexplore.ieee.org/iel5/4058334/4058335/04059285.pdf

[82] T. Matsumaru, K. Hagiwara, and T. Ito, "Examination on the combination control of manual operation and autonomous motion for teleoperation of mobile robot using a software simulation," *Transactions of the Society of Instrument and Control Engineers*, vol. 41, pp. 157–166, 2005.

[83] L. Matthies and A. Elfes, "Integration of sonar and stereo range data using a grid-based representation," in *IEEE International Conference on Robotics and Automation*, vol. 2, Apr 1988, pp. 727–733. [Online]. Available: http://ieeexplore.ieee.org/iel4/202/541/00012145.pdf

[84] H. Miura, T. Yasuda, Y. K. Fujisawa, Y. Kuwana, S. Takeuchi, and I. Shimoyama, "What is robot intelligence?" in *IEEE IECON 22nd International Conference on Industrial Electronics, Control, and Instrumentation*, 1996. [Online]. Available: http://ieeexplore.ieee.org/iel3/4195/12281/00570884.pdf

[85] K. Miyazawa, "Fire robots developed by the tokyo fire department," *Advanced Robotics*, vol. 16, no. 6, pp. 553–556, 2002.

[86] L. S. Monteiro, T. Moore, and C. J. Hill, "What is the accuracy of dgps?" *Journal of Navigation*, vol. 58, no. 2, pp. 207–225, 2005.

[87] M. Montemerlo and S. Thrun, "Simultaneous localization and mapping with unknown data association using fastslam," in *International Conference on Robotics and Automation*, vol. 2, 2003, pp. 1985– 1991. [Online]. Available: http://ieeexplore.ieee.org/iel5/8794/27834/01241885.pdf

[88] K. Mühlmann, D. Maier, J. Hesser, and R. Männer, "Calculating dense disparity maps from color stereo images, an efficient implementation," *International Journal of Computer Vision*, vol. 47, no. 1, pp. 79–88, Apr 2002. [Online]. Available: http://www.springerlink.com/media/dlycywuvtm0j181kvrt0/contributions/9/a/5/b/9a5bqnqblh8a3pk1.pdf

[89] B. Muirhead, "Mars rovers, past and future," in *IEEE Aerospace Conference*, 2004. [Online]. Available: http://ieeexplore.ieee.org/iel5/9422/29900/01367598.pdf

[90] S. Murata, E. Yoshida, K. Tomita, H. Kurokawa, A. Kamimura, and S. Kokaji, "Hardware design of modular robotic system," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000, pp. 2210–2217. [Online]. Available: http://ieeexplore.ieee.org/iel5/7177/19356/00895297.pdf

[91] R. Murphy, "Marsupial and shape-shifting robots for urban search and rescue," *IEEE Intelligent Systems*, vol. 15, no. 2, pp. 14–19, Mar 2000. [Online]. Available: http://ieeexplore.ieee.org/iel5/5254/18496/00850822.pdf

[92] ——, *Introduction to AI Robotics*, ser. Intelligent Robots and Autonomous Agents. The MIT Press, 2000.

[93] R. Murphy, J. Casper, J. Hyams, M. Micire, and B. Minten, "Mobility and sensing demands in usar," in *26th Annual Conference of the IEEE Industrial Electronics Society*, 2000, pp. 138–142. [Online]. Available: http://ieeexplore.ieee.org/iel5/7662/20973/00973139.pdf

[94] R. Murphy, J. Casper, M. Micire, and J. Hyams, "Assessment of the nist standard test bed for urban search and rescue," in *NIST Workshop on Performance Metrics for Intelligent Systems 2000*, 2000. [Online]. Available: http://crasar.csee.usf.edu/robotics/Publications/nist.pdf

[95] R. Murphy, J. Blitch, and J. Casper, "Aaai/robocup-2001 urban search and rescue events: Reality and competition," *AI Magazine*, vol. 23, no. 1, pp. 37–42, 2002. [Online]. Available: http://crasar.csee.usf.edu/robotics/Publications/nist.doc

[96] D. Murray and C. Jennings, "Stereo vision based mapping and navigation for mobile robots," in *IEEE International Conference on Robotics and Automation*, 1997. [Online]. Available: http://ieeexplore.ieee.org/iel3/4815/13374/00614387.pdf

[97] K. Nagatani, S. Tachibana, M. Sofne, and Y. Tanaka, "Improvement of odometry for omnidirectional vehicle using optical flow information," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 2000, pp. 468–473. [Online]. Available: http://ieeexplore.ieee.org/iel5/7177/19350/00894648.pdf

[98] H. Nakai, N. Takeda, H. Hattori, Y. Okamoto, and K. Onoguchi, "A practical stereo scheme for obstacle detection in automotive use," in *17th International Conference on Pattern Recognition*, 2004. [Online]. Available: http://ieeexplore.ieee.org/iel5/9258/29387/01334538.pdf

[99] National Space Science Data Center, "Luna 17/lunokhod 1," 2006, visited 2007. [Online]. Available: http://nssdc.gsfc.nasa.gov/nmc/tmp/1970-095A.html

[100] M. Negnevistsky, *Artificial Intelligence: A Guide to Intelligent Systems*. Addison-Wesley, 2002.

[101] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, June 2004, pp. 652–659. [Online]. Available: http://ieeexplore.ieee.org/iel5/9183/29133/01315094.pdf

[102] M. Okutomi and T. Kanade, "A multiple-baseline stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 4, pp. 353–363, 1993. [Online]. Available: http://www.ri.cmu.edu/pub_files/pub2/okutomi_m_1993_1/okutomi_m_1993_1.pdf

[103] C. F. Olson and H. Abi-Rached, "Wide-baseline stereo experiments in natural terrain," in *12th International Conference on Advanced Robotics*, 2005. [Online]. Available: http://ieeexplore.ieee.org/iel5/10070/32295/01507438.pdf

[104] N. Padhy, *Artificial Intelligence and Intelligent Systems*. Oxford University Press, 2005.

[105] Y. S. Park, K. H. Kang, T. F. Ewing, E. L. Faulting, B. P. DeJong, M. A. Peshkin, and J. E. Colgate, "Semi-autonomous telerobotic manipulation: a viable approach for space structure deployment and maintenance," in *AIP Conference Proceedings*, 2006.

[106] Parks Victoria, "Churchill national park & lysterfield park - visitor guide," 2006, visited 2006. [Online]. Available: http://www.parkweb.vic.gov.au/resources05/05_0310.pdf

[107] P. Payeur, "Improving robot path planning efficiency with probabilistic virtual environment models," in *IEEE Symposium on Virtual Environments, Human-Computer Interfaces and Measurement Systems*, 2004. [Online]. Available: http://ieeexplore.ieee.org/iel5/9613/30386/01397177.pdf

[108] L. C. A. Pimenta, A. R. Fonseca, G. A. S. Pereira, R. C. Mesquita, E. J. Silva, W. M. Caminhas, and M. F. M. Campos, "Robot navigation based on electrostatic field computation," *IEEE Transactions on Magnetics*, vol. 42, pp. 1459–1462, 2006. [Online]. Available: http://ieeexplore.ieee.org/iel5/20/33780/01608492.pdf

[109] Point Grey Research Inc., "Products and services: Stereo vision products: Bumblebee 2," 2006, visited 2006. [Online]. Available: http://www.ptgrey.com/products/bumblebee2/index.asp

[110] ——, "Products and services: Stereo vision products: Digiclops," 2006, visited 2006. [Online]. Available: http://www.ptgrey.com/products/digiclops/index.asp

[111] C. Poynton, *A Technical Introduction to Digital Video*. John Wiley & Sons, 1996. [Online]. Available: http://www.poynton.com/notes/TIDV/index.html

[112] ——, "Color faq - frequently asked questions color," 2006, visited 2006. [Online]. Available: http://www.poynton.com/ColorFAQ.html

[113] E. Prassler, E. Stroulia, and M. Strobel, "Office waste cleanup: an application for service robots," in *IEEE International Conference on Robotics and Automation*, 1997.

[114] C. Rasmussen, "Combining laser range, color, and texture cues for autonomous road following," in *IEEE International Conference on Robotics and Automation*, vol. 4, 2002, pp. 4320–4325. [Online]. Available: http://ieeexplore.ieee.org/iel5/7916/21828/01014439.pdf

[115] ——, "Grouping dominant orientations for ill-structured road following," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2004, pp. 470–477. [Online]. Available: http://ieeexplore.ieee.org/iel5/9183/29133/01315069.pdf

[116] D. Raviv and M. Herman, "Visual control signals for road following," in *IEEE International Symposium on Intelligent Control*, Aug 1991, pp. 436–442. [Online]. Available: http://ieeexplore.ieee.org/iel2/571/4770/00187397.pdf

[117] R. Rogers, "Improved heading using dual speed sensors for angular rate and odometry in land navigation," in *IEEE 1998 Position Location and Navigation Symposium*, Apr 1998, pp. 20–23. [Online]. Available: http://ieeexplore.ieee.org/iel4/5490/14769/00670039.pdf

[118] S. J. Russell and P. Norvig, *Artificial intelligence : a modern approach*. Prentice Hall, 2003.

[119] P. Rybski, N. Papanikolopoulos, S. Stoeter, D. Krantz, K. Yesin, M. Gini, R. Voyles, D. Hougen, B. Nelson, and M. Erickson, "Enlisting rangers and scouts for reconnaissance and surveillance," *IEEE Robotics & Automation Magazine*, vol. 7, no. 4, pp. 14–24, Dec 2000. [Online]. Available: http://ieeexplore.ieee.org/iel5/100/19341/00894029.pdf

[120] P. Rybski, S. Stoeter, M. Gini, D. Hougen, and N. Papanikolopoulos, "Performance of a distributed robotic system using shared communications channels," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 713–727, Oct 2002. [Online]. Available: http://ieeexplore.ieee.org/iel5/70/22928/01067993.pdf

[121] Y. Saab and M. VanPutte, "Shortest path planning on topographical maps," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 29, pp. 139–150, 1999. [Online]. Available: http://ieeexplore.ieee.org/iel4/3468/15875/00736370.pdf

[122] M. Saptharishi, C. S. Oliver, C. Diehl, K. Bhat, J. Dolan, A. Trebi-Ollennu, and P. Khosla, "Distributed surveillance and reconnaissance using multiple autonomous atvs: Cyberscout," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 826–836, Oct 2002. [Online]. Available: http://ieeexplore.ieee.org/iel5/70/22928/01068001.pdf

[123] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1–3, pp. 7–42, 2002. [Online]. Available: http://www.breadnet.middlebury.edu/stereo/taxonomy-IJCV.pdf

[124] G. Schweitzer, "What do we expect from intelligent robots?" in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1999, pp. 1271–1278. [Online]. Available: http://ieeexplore.ieee.org/iel5/6568/17542/00811655.pdf

[125] N. Sebe and M. S. Lew, *Robust computer vision : theory and applications*. Kluwer Academic Publishers, 2003.

[126] Sony Corporation, "Qrio," 2006, visited 2007. [Online]. Available: http://www.sony.net/SonyInfo/QRIO/

[127] D. Spero, "A review of outdoor robotics research," Monash University, Tech. Rep., 2004. [Online]. Available: http://www.ds.eng.monash.edu.au/techrep/reports/2004/MECSE-17-2004.pdf

[128] D. Spero and R. Jarvis, "Towards exteroceptive based localisation," in *Proceeding of the 2004 IEEE Conference on Robotics, Automation and Mechatronics*, 2004, pp. 822–827.

[129] D. J. Spero, "Simultaneous localisation and map building: The kidnapped way," Ph.D. dissertation, Intelligent Robotics Research Centre, Monash University, 2005. [Online]. Available: http://www.dorianspero.com/publications/Spero05b.pdf

[130] Standford Racing Team, "Standford racing team," 2005, visited 2006. [Online]. Available: http://cs.stanford.edu/group/roadrunner//old/index.html

[131] H. Stone, "Mars pathfinder microrover: A low-cost, low-power spacecraft," in *AIAA Forum on Advanced Developments in Space Robotics*, 1996.

[132] H. Stone and G. Edmonds, "Hazbot: a hazardous materials emergency response mobile robot," in *IEEE International Conference on Robotics and Automation*, May 1992, pp. 67–73. [Online]. Available: http://ieeexplore.ieee.org/iel2/399/5764/00220333.pdf

[133] C. Sun, "Fast stereo matching using rectangular subregioning and 3d maximum-surface techniques," *International Journal of Computer Vision*, vol. 47, pp. 99–117, 2002. [Online]. Available: http://extra.cmis.csiro.au/IA/changs/doc/sun02ijcv.pdf

[134] H. Sunyoto, W. van der Mark, and D. M. Gavrila, "A comparative study of fast dense stereo vision algorithms," in *IEEE Intelligent Vehicles Symposium*, 2004. [Online]. Available: http://ieeexplore.ieee.org/iel5/9278/29469/01336402.pdf

[135] The Aerospace Corporation, "Gps primer," 2003, visited 04/2005. [Online]. Available: http://www.aero.org/education/primers/gps/GPS-Primer.pdf

[136] The White House, "Statement by the president regarding the united states' decision to stop degrading global positioning system accuracy," 2000, visited 2007. [Online]. Available: http://www.ngs.noaa.gov/FGCS/info/sans_SA/docs/statement.html

[137] C. Thorpe, M. Herbert, T. Kanade, and S. Shafer, "Toward autonomous driving: the cmu navlab. i. perception," *IEEE Expert*, vol. 6, no. 4, pp. 31–42, 1991. [Online]. Available: http://ieeexplore.ieee.org/iel3/64/2809/00085919.pdf

[138] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," *Artificial Intelligence*, vol. 99, no. 1, pp. 21–71, 1998. [Online]. Available: http://www.cs.cmu.edu/~thrun/papers/thrun.map_learning.ps.gz

[139] ——, "When robots meet people," *IEEE Intelligent Systems and Their Applications*, vol. 13, pp. 27–29, 1998. [Online]. Available: http://ieeexplore.ieee.org/iel4/5254/15016/00683178.pdf

[140] S. Thrun, M. Bennewitz, W. W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Minerva: a second-generation museum tour-guide robot," in *IEEE International Conference on Robotics and Automation*, 1999. [Online]. Available: http://ieeexplore.ieee.org/iel5/6243/16696/00770401.pdf

[141] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, Cambridge, Massachusetts, 2005.

[142] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Winning the darpa grand challenge," *Journal of Field Robotics*, 2006, accepted for publication. [Online]. Available: http://robots.stanford.edu/papers/thrun.stanley05.pdf

[143] N. Tomatis, I. Nourbakhsh, and R. Siegwart, "Simultaneous localization and map building: a global topological model with local metric maps," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001. [Online]. Available: http://ieeexplore.ieee.org/iel5/7677/20976/00973393.pdf

[144] P. Trahanias, W. Burgard, A. Argyros, D. Hahnel, H. Baltzakis, P. Pfaff, and C. Stachniss, "Tourbot and webfair: Web-operated mobile robots for tele-presence in populated exhibitions," *IEEE Robotics & Automation Magazine*, vol. 12, pp. 77–89, 2005. [Online]. Available: http://ieeexplore.ieee.org/iel5/100/31383/01458329.pdf

[145] A. Trebi-Ollennu and J. Dolan, "An autonomous ground vehicle for distributed surveillance: Cyberscout," ICES, Carnegie Mellon University, Tech. Rep. ICES-04-09-99, July 1999. [Online]. Available: http://www.ri.cmu.edu/pub_files/pub2/trebi_ollennu_ashitey_1999_3/trebi_ollennu_ashitey_1999_3.pdf

[146] G. Trimble, "Autonomous operation of the explosive ordnance disposal robotic work package using the cetus untethered underwater vehicle," in *Symposium on Autonomous Underwater Vehicle Technology*, 1996, pp. 21–27. [Online]. Available: http://ieeexplore.ieee.org/iel3/3780/11039/00532396.pdf

[147] K. I. Trovato and L. Dorst, "Differential a*," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, pp. 1218–1229, 2002. [Online]. Available: http://ieeexplore.ieee.org/iel5/69/22458/01047763.pdf

[148] E. Trucco and A. Verri, *Introductory techniques for 3-D computer vision*. Prentice Hall, 1998.

[149] R. J. Turner, "Slip measurement using dual radar guns," in *International Summer Meeting of THE AMERICAN SOCIETY OF AGRICULTURAL ENGINEERS and THE CANADIAN SOCIETY OF AGRICULTURAL ENGINEERING*, 1993. [Online]. Available: http://www1.agric.gov.ab.ca/$department/deptdocs.nsf/all/eng8286

[150] Turner-Fairbank Highway Research Center, "Nationwide differential global positioning system program fact sheet," 2007, visited 2007. [Online]. Available: http://www.tfhrc.gov/its/ndgps/02072.htm

[151] C. Urmson, J. Anhalt, M. Clark, T. Galatali, J. Gonzalez, J. Gowdy, A. Gutierrez, S. Harbaugh, M. Johnson-Roberson, H. Kato, P. Koon, K. Peterson, B. .Smith, S. Spiker, E. Tryzelaar, and W. Whittaker, "High speed navigation of unrehearsed terrain: Red team technology for grand challenge 2004," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-04-37, June 2004. [Online]. Available: http://www.ri.cmu.edu/pub_files/pub4/urmson_christopher_2004_1/urmson_christopher_2004_1.pdf

[152] O. Veksler, "Stereo correspondence by dynamic programming on a tree," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005. [Online]. Available: http://www.csd.uwo.ca/faculty/olga/Papers/cvpr05.pdf

[153] K. Wedeward, S. Bruder, T. Yodaiken, and J. Guilberto, "Low-cost outdoor mobile robot: A platform for landmine detection," in *42nd Midwest Symposium on Circuits and Systems*, 1999, pp. 131–134. [Online]. Available: http://ieeexplore.ieee.org/iel5/6968/18758/00867226.pdf

[154] R. Wegner and J. Anderson, "Agent-based support for balancing teleoperation and autonomy in urban search and rescue," *International Journal of Robotics and Automation*, vol. 21, pp. 120–127, 2006.

[155] M. Williamson, "Man on the moon: the technology of lunar exploration," *Engineering Science and Education Journal*, vol. 11, pp. 217–226, 2002. [Online]. Available: http://ieeexplore.ieee.org/iel5/2222/26024/01161557.pdf

[156] G. Xú and Z. Zhang, *Epipolar geometry in stereo, motion, and object recognition : a unified approach*. Kluwer Academic Publishers, 1996.

[157] M. Yim, D. Duff, and K. Roufas, "Polybot: a modular reconfigurable robot," in *IEEE International Conference on Robotics and Automation*, 2000, pp. 514–520. [Online]. Available: http://ieeexplore.ieee.org/iel5/6794/18235/00844106.pdf

[158] C. Yinger, "Operation and application of the global positioning system," *The Aerospace Press: Crosslink*, vol. 3, no. 2, pp. 12–16, 2002. [Online]. Available: http://www.aero.org/publications/crosslink/pdfs/CrosslinkV3N2.pdf

[159] C.-H. Yu and F. Caimi, "Determination of horizontal motion through optical flow computations," in *OCEANS '93. 'Engineering in Harmony with Ocean'*, vol. 2, Oct 1993, pp. 475–480. [Online]. Available: http://ieeexplore.ieee.org/iel2/1081/7739/00326142.pdf

[160] D. Yuen and B. MacDonald, "An evaluation of the sequential monte carlo technique for simultaneous localisation and map-building," in *IEEE International Conference on Robotics and Automation*, vol. 2, 2003, pp. 1564–1569. [Online]. Available: http://ieeexplore.ieee.org/iel5/8794/27834/01241817.pdf

[161] A. Zelinsky, "A mobile robot exploration algorithm," *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 707–717, 1992. [Online]. Available: http://ieeexplore.ieee.org/iel4/70/4677/00182671.pdf

[162] J. Zhang and H. Nagel, "Texture-based segmentation of road images," in *Intelligent Vehicles '94 Symposium*, 1994, pp. 260–265. [Online]. Available: http://ieeexplore.ieee.org/iel3/5064/13852/00639516.pdf

Appendices

The Geometry of Stereo Vision: An Overview

The geometry and corresponding mathematics describing a standard stereo system have been covered in great detail in a number of publications such as Howard and Rogers [56], Xú and Zhang [156], Trucco and Verri [148] and Sebe and Lew [125]. Accordingly, only a minimal overview will be presented here.

The material presented here largely follows the mathematical (and particularly geometric) conventions used in the "*Camera Calibration Toolbox for Matlab*" [10].

.1 Two Cameras, Two Images

As shown in Figure 3 we have two cameras producing two images, labelled as image_L and image_R for the left and right cameras, respectively. A point $\boldsymbol{P}_L$ is projected onto the left image as $\boldsymbol{p}_L = [x_L, y_L, f_L]^T$ where f_L is the focal length. Similarly, a point $\boldsymbol{P}_R$ is projected onto the right image as $\boldsymbol{p}_R = [x_R, y_R, f_R]^T$. Most often, $f_L = f_R = f$, that is, focal lengths are the same and simply represented by f.

The points $\boldsymbol{p}_L$ and $\boldsymbol{p}_R$ are related by

$$\boldsymbol{p}_R = \mathsf{R}_{ext}\boldsymbol{p}_L + \boldsymbol{T}_{ext} \tag{1}$$

where R_{ext} is a rotation matrix and $\boldsymbol{T}_{ext}$ a translation vector that describe the difference in orientation and position of the two cameras. (Note that, by this convention, the x-component of $\boldsymbol{T}_{ext}$ will be negative.) These are referred to as the extrinsic parameters of the stereo system. This can alternatively be represented by the inverse relationship

$$\boldsymbol{p}_L = \mathsf{R}_{ext}^{-1}(\boldsymbol{p}_R - \boldsymbol{T}_{ext}) \tag{2}$$

.2 Epipolar Geometry

Figure 4 shows further details of the geometry of stereo – known as epipolar geometry. In the left reference frame, $\boldsymbol{e}_L$ is the projection of the right camera's centre of projection ($\boldsymbol{O}_R$) onto the left image. Similarly, $\boldsymbol{e}_R$ is the projection of the left camera's centre of projection ($\boldsymbol{O}_L$) onto the right image. These points are referred to as *epipoles*. An arbitrary point $\boldsymbol{P}$ seen by both left and right cameras creates a triangle (and hence defines a plane, known as an epipolar plane) with corners at $\boldsymbol{P}$ and the two centres of projection $\boldsymbol{O}_L$ and $\boldsymbol{O}_R$. The base of this triangle, $\overrightarrow{O_L O_R}$ passes through the epipoles. Indeed, the triangle and hence epipolar

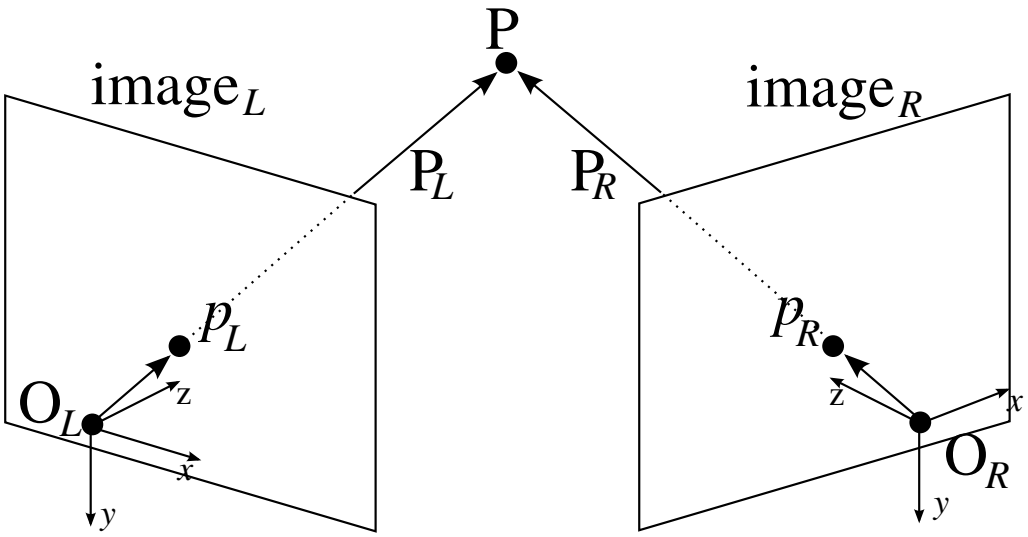


Figure 3: Stereo geometry

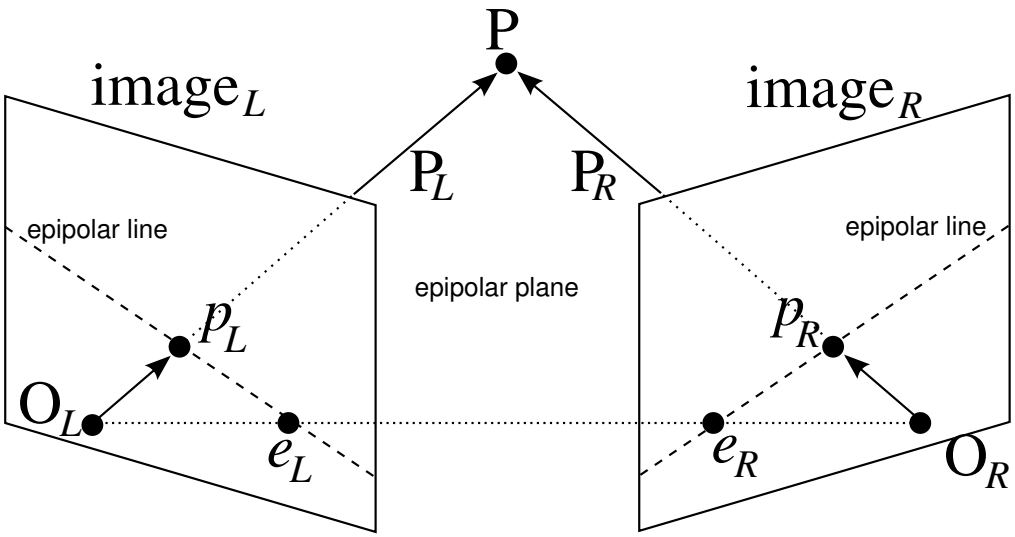


Figure 4: Epipolar geometry

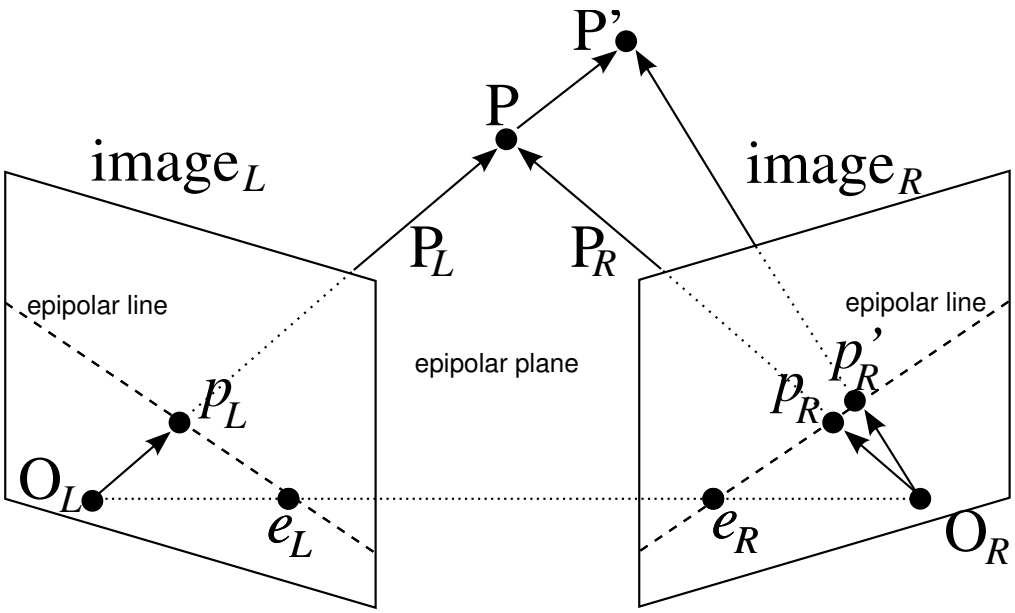


Figure 5: The epipolar constraint as applied to correspondence

plane defined by *any* arbitrary point includes this same base, and so *all* epipolar planes pass through the epipoles.

The epipolar plane is projected onto the left image as an epipolar line running through $\boldsymbol{p}_L$ and $\boldsymbol{e}_L$ and onto the right image as an epipolar line passing through $\boldsymbol{p}_R$ and $\boldsymbol{e}_R$. Because the epipolar lines lie on the epipolar plane, all epipolar lines also must pass through the corresponding left or right epipole.

.3 The Epipolar Search Constraint

The importance of epipolar geometry to the stereo problem is apparent when considering the task of finding correspondences, that is, matching a feature in the left image with a feature in the right image. In Figure 5 the point seen in the left image at $\boldsymbol{p}_L$ could correspond to points in the right image at either $\boldsymbol{p}_R$ or $\boldsymbol{p}_R$', depending on whether the point actually lies at $\boldsymbol{P}$ or $\boldsymbol{P}'$. However, $\boldsymbol{P}$ and $\boldsymbol{P}'$ (or indeed *any* point that projects onto $\boldsymbol{p}_L$) define the same epipolar plane and hence the same epipolar lines in the left and right images. Therefore, the search in the right image for the point corresponding to $\boldsymbol{p}_L$ is constrained to a search along the right-image epipolar line associated with epipolar plane defined by $\boldsymbol{p}_L$. Knowledge of the epipolar geometry of a stereo rig provides a correspondence search constraint that significantly reduces the search space. As a result, a first stage in any stereo system is the rectification of images in order to align epipolar lines, both left-to-right and with image scanlines, thus simplifying the correspondence search process.

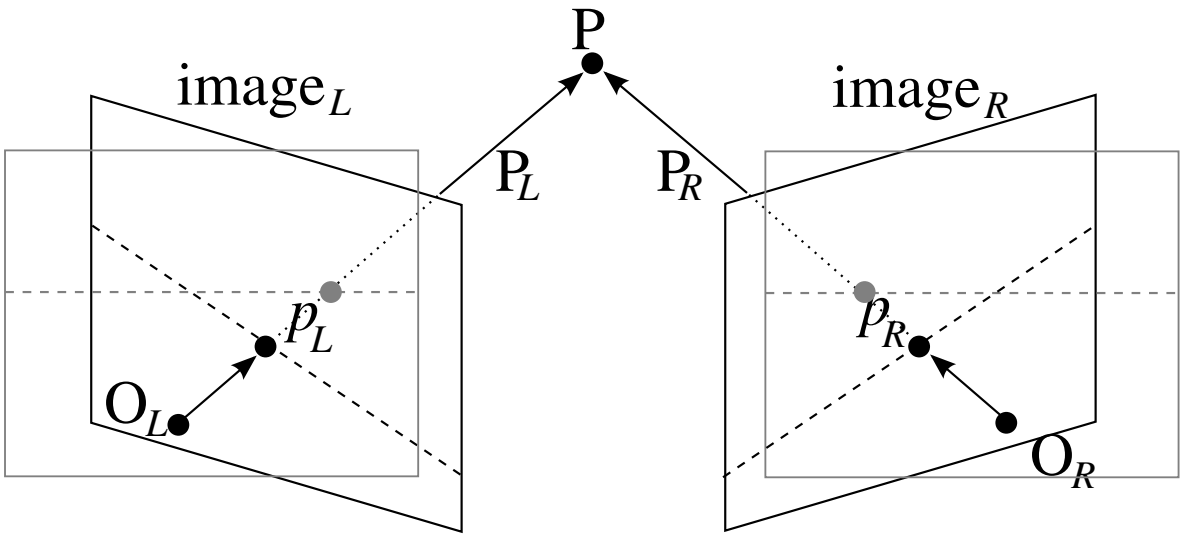


Figure 6: Rectification of stereo images

.4 Image Rectification

Image rectification, in the context of stereo vision, is the process of remapping pairs of images (left and right), in order to take advantage of the epipolar constraint. As depicted in Figure 6, rectification maps the images such that epipolar lines are made to correspond with scanlines (image rows) and usually also with each-other, that is, the epipolar line on row n of the left image corresponds with the epipolar line on row n of the right image. When searching for feature or point correspondence, we then simply search in one dimension along the corresponding image row in the search image (that is, with a point in the left, we look along the same row in the right image). This leads to significant computational savings.

.5 Determination of Depth

At this point, we make the assumption that we know the extrinsic parameters of the stereo rig and we also have solved the correspondence problem: for a point in the left image $\boldsymbol{p}_L$ we have the corresponding point in the right image $\boldsymbol{p}_R$. We now look at how we can determine the three-dimensional position of the point, with respect to the left (reference) camera.

As shown in Figure 7, $\boldsymbol{p}_L$ and $\boldsymbol{p}_R$ define two lines, $a\boldsymbol{p}_L$ and $b\boldsymbol{p}_R$ parameterised by the scalars a and b respectively. Ideally, these lines would intersect at $\boldsymbol{P}$, however, because $\boldsymbol{p}_L$ and $\boldsymbol{p}_R$ are known only approximately (in the very least because of pixel quantisation), the lines invariably do not intersect. Following Trucco and Verri [148] we therefore resolve $\boldsymbol{P}$ as the point where the lines are closest.

Let $\boldsymbol{w} = \boldsymbol{p}_L \times \mathsf{R}_{ext}^{-1}(\boldsymbol{p}_R - \boldsymbol{T}_{ext})$ be a vector (in the left camera's reference frame) orthogonal to both $\boldsymbol{p}_L$ and $\boldsymbol{p}_R$. We wish then to find the midpoint of the line segment parallel to $\boldsymbol{w}$ and joining $a\boldsymbol{p}_L$ and $b\boldsymbol{p}_R$.

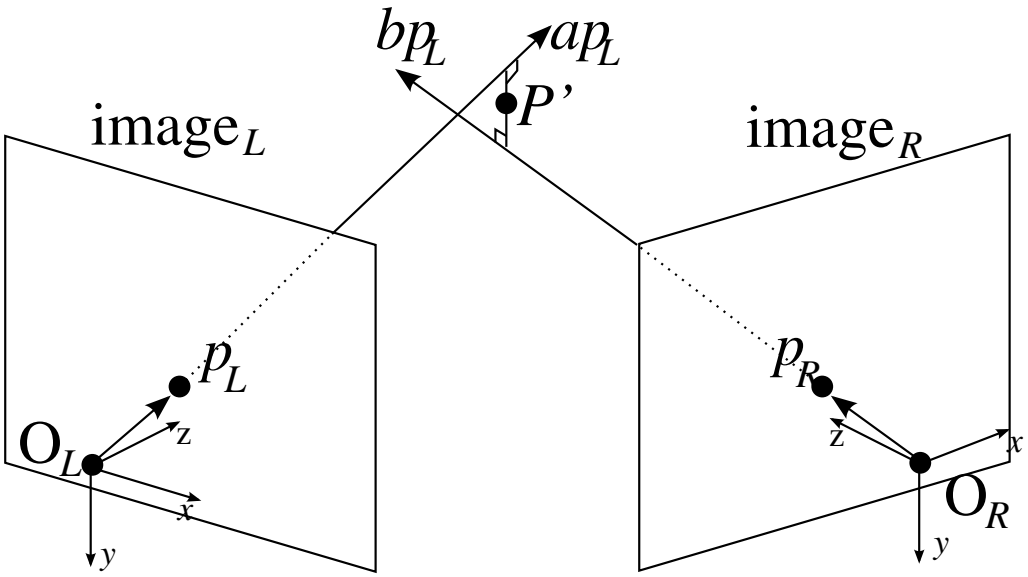


Figure 7: Determination of depth

Let $a_0\boldsymbol{p}_L$ and $b_0\boldsymbol{p}_R$ be the endpoints of the line segment $c_0\boldsymbol{w}$. We then have

$$\begin{aligned} & a_0\boldsymbol{p}_L + c_0\boldsymbol{w} - \mathsf{R}_{ext}^{-1}(b_0\boldsymbol{p}_R - \boldsymbol{T}_{ext}) = 0 \\ \implies & a_0\boldsymbol{p}_L - b_0\mathsf{R}_{ext}^{-1}\boldsymbol{p}_R + c_0\boldsymbol{p}_L \times \mathsf{R}_{ext}^{-1}(\boldsymbol{p}_R - \boldsymbol{T}_{ext}) = -\mathsf{R}_{ext}^{-1}\boldsymbol{T}_{ext} \end{aligned} \tag{3}$$

With $\boldsymbol{p}_L$, $\boldsymbol{p}_R$, R_{ext} and $\boldsymbol{T}_{ext}$ known, the three-dimensional position of $\boldsymbol{P}$ can then be found by solving Eq^n 3 for a_0, b_0 and c_0 and finding the midpoint of the segment joining $a_0\boldsymbol{p}_L$ and $b_0\boldsymbol{p}_R$.

For completeness (and comparison with commonly presented stereo equations) we will look at a simplified common case solution. Firstly, we assume that $c_0 = 0$ and $\boldsymbol{P} = a_0\boldsymbol{p}_L$, that is, the two lines intersect exactly (no errors or approximations). Secondly, we assume rectified images, so that $\mathsf{R}_{ext} = \mathsf{R}_{ext}^{-1} = \mathsf{I}$ (no relative rotation between cameras), $\boldsymbol{T}_{ext} = [-B, 0, 0]$ (the translation between cameras is purely in the x-direction and negative by convention of Eq^n 1) and $\boldsymbol{p}_R = \boldsymbol{p}_L - \boldsymbol{d}$ with $\boldsymbol{d} = [x_d, 0, 0]^T$ (epipolar lines are aligned, so that the right image point is the same as the left image point except for an offset purely in the negative x-direction). Finally, we assume $\boldsymbol{p}_L = [x_L, y_L, f]^T$, where f is the common focal length.

We then have

$$\begin{aligned} & a_0\boldsymbol{p}_L - b_0\mathsf{R}_{ext}^{-1}\boldsymbol{p}_R + c_0\boldsymbol{p}_L \times \mathsf{R}_{ext}^{-1}(\boldsymbol{p}_R - \boldsymbol{T}_{ext}) = -\mathsf{R}_{ext}^{-1}\boldsymbol{T}_{ext} \\ \implies & a_0\boldsymbol{p}_L - b_0(\boldsymbol{p}_L - \boldsymbol{d}) = -\boldsymbol{T}_{ext} \\ \implies & (a_0 - b_0)\boldsymbol{p}_L + b_0\boldsymbol{d} = -\boldsymbol{T}_{ext} \end{aligned}$$

giving

$$(a_0 - b_0)x_L + b_0 x_d = B \tag{4}$$
$$(a_0 - b_0)y_L = 0 \tag{5}$$
$$(a_0 - b_0)f = 0 \tag{6}$$

From Eq^n 6, with $f \neq 0$, we have $a_0 = b_0$ and therefore from Eq^n 4

$$b_0 x_d = B$$
$$\implies b_0 = a_0 = \frac{B}{x_d}$$

We then have, in the left camera's reference frame, $\boldsymbol{P} = a_0 \boldsymbol{p}_L$ and the depth of this point, call it Z, is then

$$Z = a_0 z_L$$
$$\implies Z = \frac{B}{x_d} f \tag{7}$$

which is the standard depth-resolution equation presented in most if not all textbooks on the subject. The derivation given here is different to the typical derivation, which *begins* with simplifying assumptions and then builds the geometry and mathematics using the principal of similar triangles.

Verbally, Eq^n 7 states that the depth of the point is proportional to the separation of cameras B, known as the baseline, and inversely proportional to the difference in location of the point between the left and right images x_d, known as the disparity. In general, the disparity is proportional to the distance of the point from the location of camera convergence. In this example, with no rotation between cameras, the cameras do not converge, or alternatively, the point of convergence is at infinity, hence the inversely proportional relationship between disparity and distance from the camera.

.6 The Effect of Camera Configuration

The material presented thus-far assumes a pair of cameras, operating as a stereo pair. These principals also extend to more than two cameras that can be used together (typically in pairs) for improved results. A common variation uses three cameras – known as trinocular vision – with three cameras forming a right angle. These are used as two stereo pairs, one aligned horizontally and one aligned vertically. This can provide redundancy that can result in higher reliability. The Point Grey Research® Inc. Digiclops® [110] system is one commercial example of a trinocular vision system and is shown in Figure 8. Alternative configurations use collinear cameras (that is, all baselines fall upon the same line) and make use of multiple baselines (for example [102]).

Figure 9 shows the effect of baseline on disparity. For a given point, a wider baseline ($B' > B$) produces a greater disparity ($d' > d$). Because disparity is measured from an image, there is a definite minimum measurable disparity (typically a disparity of one pixel, although techniques exist to estimate sub-pixel disparities). With disparity being inversely proportional to distance, this minimum disparity corresponds to a maximum range. Therefore,

Figure 8: The Point Grey Research® Inc. Digiclops® trinocular system [110]

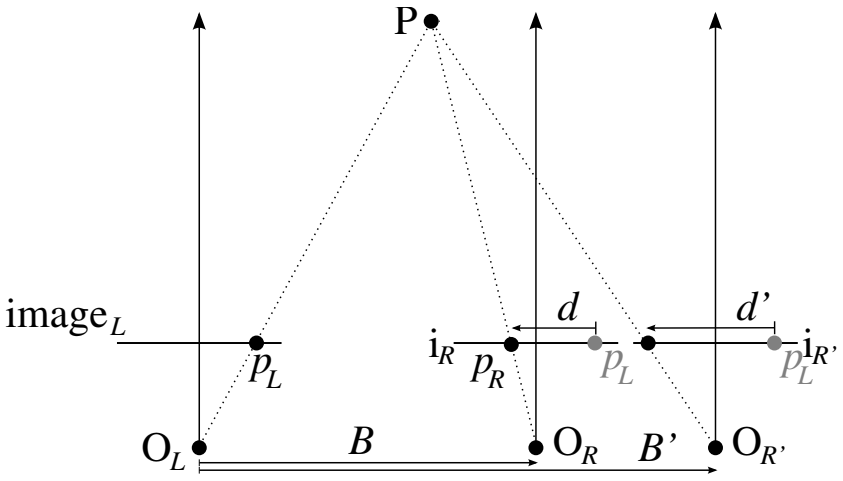


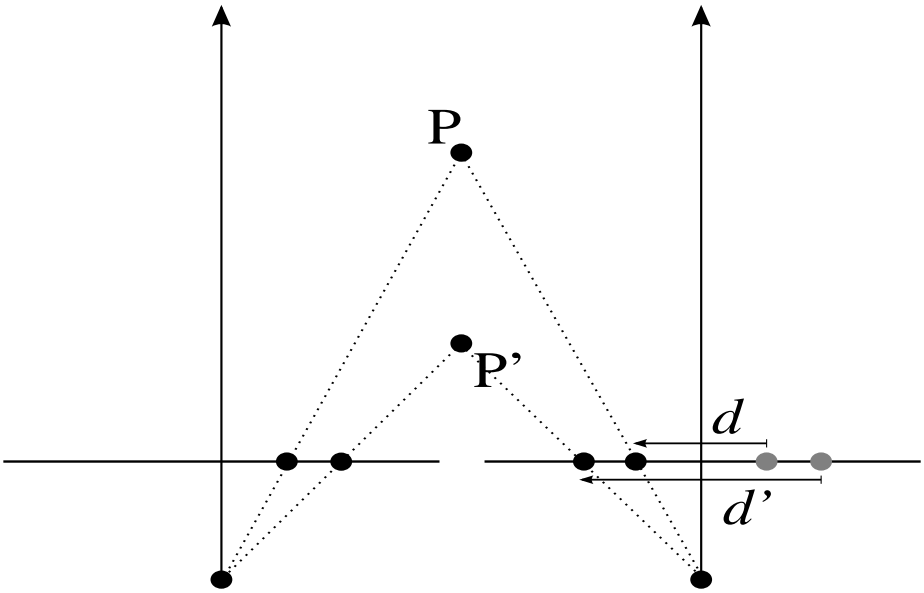
Figure 9: The effect of baseline on disparity

the baseline can be set according to the maximum range required.

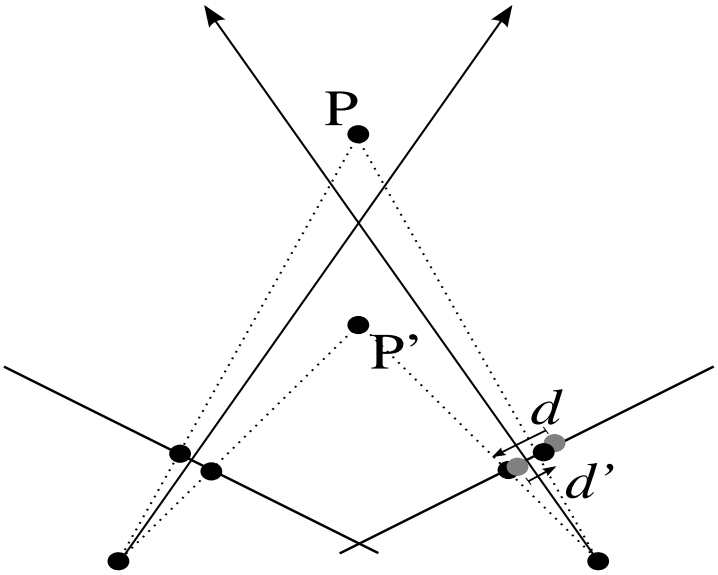
Figure 10(a) and Figure 10(b) show the effect of camera convergence on disparity. In Figure 10(a), the optical axes are parallel and the point of convergence is at infinity. As a result, the disparity is seen to decrease with increasing distance from the cameras. In Figure 10(b) the optical axes converge to a point a finite distance in front of the cameras. The disparity is seen to increase with increasing distance from the point of convergence, with a reversal in the sign of disparity depending on which side of the point of convergence (either towards the cameras or away from the cameras) the point lies.

A wide baseline is preferable because it allows for a greater measurable range and greater range resolution. There is a price, however. Figure 11 depicts the so-called *"hidden parts problem"*, where a point visible in both cameras at one baseline (B) becomes invisible to the right camera when the baseline is increased (to B'). The shaded triangles represent the fields-of-view of the cameras. When the baseline is increased, the point is no longer in the working region, that is, the region where left and right views overlap and disparity and hence depth can be measured.

Figure 11(b) and Figure 11(c) show how converging the cameras can at least partially overcome this problem. The point P' is initially out of the overlap region (indeed, it is outside either camera's field-of-view) but comes into shared view when the cameras are converged. Therefore, to partially compensate for the reduction in the working region of the rig, we can converge the cameras such that the closest point of overlap is consistent with the minimum desired range.

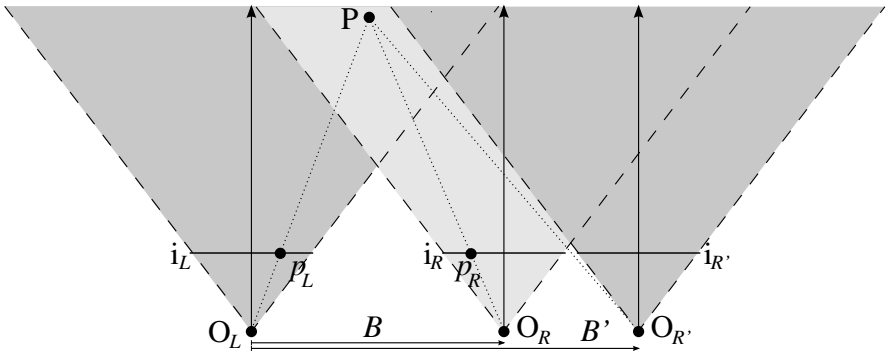


(a) Parallel optical axes

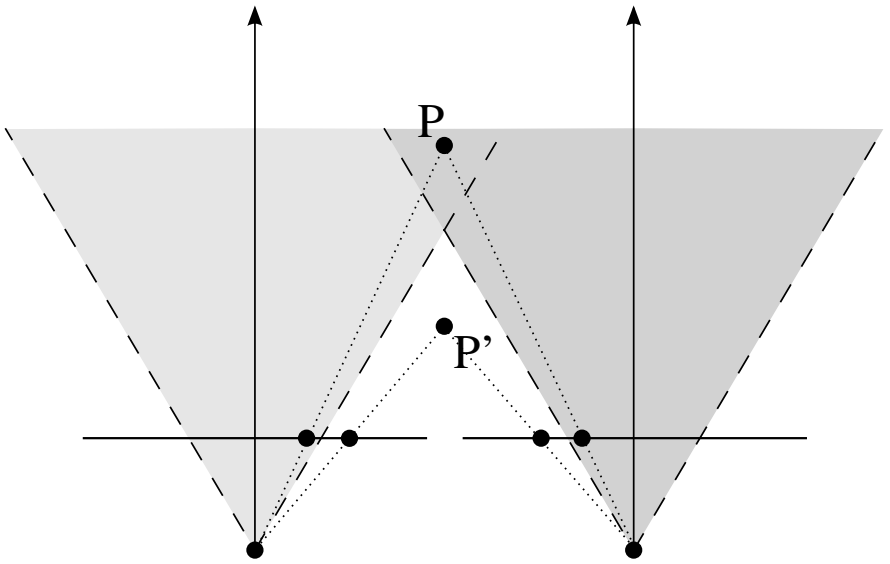


(b) Convergent optical axes

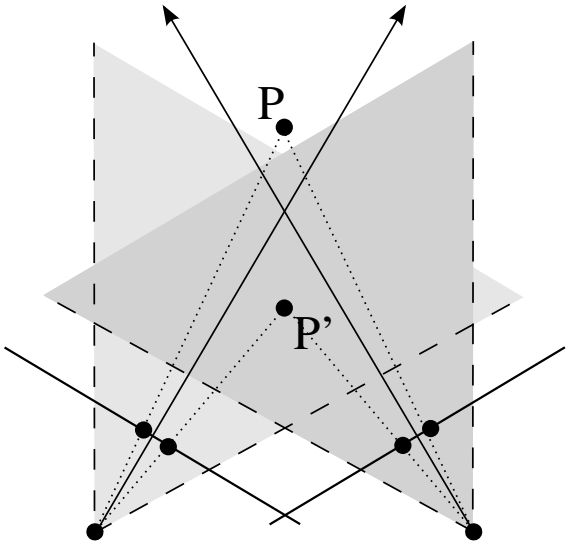
Figure 10: The effect of convergence on disparity



(a) Point hidden by increased baseline



(b) Hidden point with parallel optical axes



(c) Visible point with converging optical axes

Figure 11: The hidden parts problem

Publications

The following is a list of publications arising from the work presented in this thesis:

> D. Fernandez and A. Price, "Visual odometry for an outdoor mobile robot," in *IEEE Conference on Robotics, Automation and Mechatronics*, 2004, pp. 816–821. [Online]. Available: http://ieeexplore.ieee.org/iel5/9824/30961/01438023.pdf
> This describes an early state of the work that is presented as Chapter 3, "Relative Localisation with Visual Odometry".

> D. Fernandez and A. Price, "Visual detection and tracking of poorly structured dirt roads," in *12th International Conference on Advanced Robotics*, 2005, pp. 553–560. [Online]. Available: http://ieeexplore.ieee.org/iel5/10070/32295/01507463.pdf
> This describes an early state of the work that forms the basis of Chapters 5, "Detecting Poorly Structured Dirt Roads using Colour Vision" and 6, "Local Path-Planning using Likelihood Maps".

Electronic copies of these papers can be found on the included CD-ROM. The file `index.html` in the root directory of the CD-ROM contains an index to the additional material.

Additional Experimental Results

In addition to the results presented and discussed in Chapters 4, 5 and 6, a collection of experimental results, in the form of videos, is given on the attached CD-ROM. The file `index.html` in the root directory of the CD-ROM contains an index to the additional material.

The videos show the (image-based) output of the obstacle detection system, the road detection system and the local path-planner. During these experiments, the robot was manually driven while image sequences were captures. Processing was then performed offline.