Mobile Robot Navigation Using Active Vision

Andrew John Davison
Keble College

Robotics Research Group
Department of Engineering Science
University of Oxford

Submitted February 14 1998; Examined June 9th 1998.

This thesis is submitted to the Department of Engineering Science, University of Oxford, for the degree of Doctor of Philosophy. This thesis is entirely my own work, and, except where otherwise indicated, describes my own research.

Andrew John Davison
Keble College

Doctor of Philosophy
Hilary Term, 1998

Mobile Robot Navigation Using Active Vision

Abstract

Active cameras provide a navigating vehicle with the ability to fixate and track features over extended periods of time, and wide fields of view. While it is relatively straightforward to apply fixating vision to tactical, short-term navigation tasks, using serial fixation on a succession of features to provide global information for strategic navigation is more involved. However, active vision is seemingly well-suited to this task: the ability to measure features over such a wide range means that the same ones can be used as a robot makes a wide range of movements. This has advantages for map-building and localisation.

The core work of this thesis concerns simultaneous localisation and map-building for a robot with a stereo active head, operating in an unknown environment and using point features in the world as visual landmarks. Importance has been attached to producing maps which are useful for *extended* periods of navigation. Many map-building methods fail on extended runs because they do not have the ability to recognise previously visited areas as such and adjust their maps accordingly. With active cameras, it really is possible to re-detect features in an area previously visited, even if the area is not passed through along the original trajectory. Maintaining a large, consistent map requires detailed information to be stored about features and their relationships. This information is computationally expensive to maintain, but a sparse map of landmark features can be handled successfully. We also present a method which can dramatically increase the efficiency of updates in the case that repeated measurements are made of a single feature, permitting continuous real-time tracking of features irrespective of the total map size.

Active sensing requires decisions to be made about where resources can best be applied. A strategy is developed for serially fixating on different features during navigation, making the measurements where most information will be gained to improve map and localisation estimates. A useful map is automatically maintained by adding and deleting features to and from the map when necessary.

What sort of tasks should an autonomous robot be able to perform? In most applications, there will be at least some prior information or commands governing the required motion and we will look at how this information can be incorporated with map-building techniques designed for unknown environments. We will make the distinction between position-based navigation, and so-called context-based navigation, where a robot manoeuvres with respect to locally observed parts of the surroundings.

A fully automatic, real-time implementation of the ideas developed is presented, and a variety of detailed and extended experiments in a realistic environment are used to evaluate algorithms and make ground-truth comparisons.

Acknowledgements

First of all, I'd like to thank to my supervisor David Murray for all of his ideas, encouragement, excellent writing tips and general advice — another physics recruit successfully converted.

Next thanks must go to Ian Reid, in particular for applying his in-depth hacking skills to getting the robot hardware to work and leading the lab on its path to Linux gurudom. I'm very grateful to all the other past and present members of the Active Vision Lab who've helped daily with all aspects of my work including proof-reading, and been great company for tea breaks, pub trips, dinners, parties, hacky-sacking and all that: Jace, Torfi, John, Lourdes, Phil, Stuart, Eric, Nobuyuki, Paul, and Kevin. Thanks as well to all the other Robotics Research Group people who've been friendly, helpful, and inspiring with their high standard of work (in particular thanks to Alex Nairac for letting me use his robot's wheel and David Lee for lending me a laser).

Three and a bit more years in Oxford have been a great laugh due to the friends I've had here: house members Ian, Julie, John, Alison, Jon, Nicky, Ali, Naomi and Rebecca, frisbee dudes Siew-li, Luis, Jon, Erik, Clemens, Martin, Gill, Bobby, Jochen, Del, Derek, squash players Dave, Raja and Stephan and the rest.

Thanks a lot to my old Oxford mates now scattered around the country for weekends away, holidays and email banter: Ahrash, Phil, Rich, Vic, Louise, Clare, Ian, Amir and Tim; and to my friends from Maidstone Alex, Jon and Joji.

Last, thanks to my mum and dad and bruv Steve for the financial assistance, weekends at home, holidays, phone calls and everything else.

I am grateful to the EPSRC for funding my research.

Contents

1

Introduction

1.1 Autonomous Navigation

Progress in the field of mobile robot navigation has been slower than might have been expected from the excitement and relatively rapid advances of the early days of research [65]. As will be described later in this chapter, the most successful projects have operated in highly constrained environments or have included a certain amount of human assistance. Systems where a robot is acting independently in complicated surroundings have often been proven only in very limited trials, or have not produced actions which could be thought of as particularly "useful".

1.1.1 Why is Navigation So Difficult? (The Science "Push")

It is very interesting to consider why autonomous navigation in unknown surroundings is such a difficult task for engineered robots — after all, it is something which is taken for granted as easy for humans or animals, who have no trouble moving through unfamiliar areas, even if they are perhaps quite different from environments usually encountered.

The first thing to note is the tendency of the designers of navigation algorithms to impose methods and representations on the robot which are understandable by a human operator. Look at the inner workings of most map building algorithms for example (the one in this thesis included), and there is a strong likelihood of finding Cartesian (x, y, z) representations of the locations of features. It is not clear that this is the best way to do things on such a low level, or that biological brains have any similar kind of representation deep within their inner workings. However, having these representations does provide huge benefits when we remember that most robots exist to perform tasks which benefit humans. It makes it simple to input information which is pertinent to particular tasks, to supervise

operation when it is necessary, and, especially in the development stage of a robotic project, to understand the functioning of the algorithm and overcome any failures.

There has been a move away from this with methods such as genetic algorithms and artificial neural networks, in which no attempt is made to understand the inner workings of the processing behind a robot's actions. These methods are clearly appealing in that they seem to offer the closest analogue with the workings of the brains behind navigation systems in nature. The algorithms in a way design themselves. However, as described above, this approach creates robots which are inherently more difficult to interface with, possessing "minds of their own" to some extent.

Researchers have come up with a variety of successful modules to perform certain navigation tasks: for instance, to produce localisation information, to identify local hazards or to safely round a known obstacle. Joining these into complete systems has proved to be difficult, though. Each algorithm produces its own output in a particular form and has its own type of representation. For example, a free-space detection algorithm may generate a gridded representation of the areas filled by free space or obstacles in the vicinity of a robot: how should this be used to generate reference features to be used by a close-control technique for safe robot guidance? Alternatively, if a robot is engaged in a close-control manoeuvre, when is it time to stop and switch back to normal operation? One of the most original approaches to such problems lies with the methodology of "intelligence without representation" [14] put forward by Brooks and colleagues at MIT. In their systems, different modes of operation and sensing (behaviours) run concurrently in a layered architecture and vote towards the overall robot action while avoiding attempting to communicate representations between themselves. This will be looked at in more detail later.

The final, very important, point is that the physical construction of robots which are robust and reliable to standards anywhere approaching living bodies is extremely challenging. Problems with batteries or the restrictions imposed by an umbilical connection to an external power source in particular have meant that robots are commonly not running for long enough periods for decent evaluation of their performance and correction of their faults. Also, the complications of programming and communicating with a computer mounted on a robot and all of its devices make progress inevitably slow.

In summary, the answer to the question in the title of this section would seem to be that the complexity of the "real world", even in the simplified form of an indoor environment, is such that engineered systems have not yet come anywhere near to having the wide range of capabilities to cope in all situations. It does not appear that some kind of general overall algorithm will suffice to guide a robot in all situations, and evidence shows that this is not the case with biological navigation systems [74] which operate as a collection of specialised behaviours and tricks. Of course humans and animals, who have been subject to the process of evolution while living in the real world, have necessarily developed all the tricks required to survive.

1.1.2 Uses of Navigating Robots (The Application "Pull")

How much is there really a need for autonomous mobile robots in real-world applications? It is perhaps difficult to think of situations where autonomous capabilities are truly essential. It is frequently possible to aid the robot with human control, or to allow it to operate only under restricted circumstances. High-profile systems such as the Mars Rover or road-

following cars, which will be described later, fall into these categories.

Useful [71] mobile robots could then clearly be autonomous to different degrees, their navigational capabilities depending on the application. Consider for example a device such as an automatic vacuum cleaner or lawnmower, prototypes of both of which have been built commercially. Either of these operates in a well defined region (a particular room or lawn) and needs to be able to identify the limits of that region and avoid other potential hazards while carrying out its task. However, at least in the current designs, it does not have to have any ability to calculate its location in the region or to plan particular paths around it — the behaviour pattern of both devices is a random wandering (with the addition of an initial relatively simple edge-following stage in the case of the vacuum cleaner). In theory they could be greatly improved in terms of efficiency if they were able to measure the room or lawn and then plan a neat path to cover the area effectively — however, this would present a substantial additional challenge which is not necessarily justified. The lawnmower in particular is designed to run purely from solar power with the idea that it could run continuously over the lawn during the day (a kind of robotic sheep).

In other circumstances, it may be necessary for a robot to monitor its position accurately and execute definite movements, but possible to facilitate this in a simple way with external help such as a prior map with landmark beacons in known positions. In a factory or warehouse, where mobile robots are employed to fetch and carry, this is the obvious solution: for example, carefully placed barcode-type beacons can be scanned by a laser mounted on the robot. Robots making large movements through outdoor environments, such as unmanned full-scale vehicles, can make use of the satellite beacons of the Global Positioning System to achieve the same goal. Having this continuous knowledge of position makes the tasks of navigation much easier.

What kind of applications require a mobile robot which can navigate truly autonomously in unknown environments? Potentially, exploring remote or dangerous areas, although assistance may often still be possible here. With global maps and GPS, only other planets offer totally unknown surroundings for exploration. More realistically, in many situations where a robot could often make use of prior knowledge or external help, it is still necessary to enable it to navigate for itself in some circumstances for the system to be truly robust. The robot may need occasionally to react to changes in the environment, or move into unfamiliar parts of the territory, and take these situations in its stride as a human navigator would. A system relying solely on known beacons is vulnerable to failure if the beacons become obscured or if the robot is required to move outside of its usual operating area. Autonomous capabilities would be required when a normally known area has experienced a change — a dangerous part of a factory after an accident for example. At the Electrotechnical Laboratory in Japan the aim of such a project is to produce a robot to perform automatic inspections of a nuclear power plant. This time-consuming activity would be better performed by an autonomous robot than one controlled by a remote human operator, but it important that the robot could react in the case of a incident. The general increase in flexibility of a really autonomous robot would be beneficial in many applications. Consider the example of a robot taxi operating in a busy city: changes will happen every day, and over a longer timescale as roads are modified and landmarks alter. A human in daily contact with the city does not have to be informed of all these changes — our internal maps update themselves (although confusion can occur when revisiting a city after many years).

1.1.3 The Mars Rover

Perhaps many people's idea of the state of the art in robot navigation is the rover "Sojourner" [89] placed on Mars by the NASA Pathfinder Mission on 4th July 1997. As such, it is worth examining its operation to see what capabilities it had and which problems it was able to solve. The capsule descending onto Mars consisted of a lander module containing and cushioning the rover. Once on the planet surface, the immobile lander module acted as a base station for the rover, which carried out its explorations in the close vicinity while in regular radio contact. The rover remained operational until communications with Earth were lost on 27th September, and successfully fulfilled the main aims of the mission by collecting huge amounts of data about the surface of the planet.

The 6-wheeled rover, whose physical size is 68cm × 48cm × 28cm, was described by the mission scientists as "a spacecraft in its own right", having an impressive array of devices, sensors and components to enable it to survive in the harsh Martian conditions (where daily temperatures range from -100°C to -20°C). Its design was greatly influenced by the power restrictions imposed by the operating environment — all power had to come from the relatively small on-board solar panels (light-weight non-rechargeable batteries being present only as a backup), meaning that sufficient power was not available for simultaneous operation of various devices; in particular, the rover was not able to move while using its sensors or transmitting communications, and thus operated in a start-stop manner, switching power to where it was needed.

As the rover drove around, it was watched by high-definition stereo cameras at the lander. The robot also transmitted continuous telemetry information back to the lander about the states of all of its sensors and systems. Twice per Martian day information and pictures were forwarded from the lander to Earth. At the control station on Earth, the images from the lander were used to construct a 3D view of the rover and its surroundings from which a human operator was able to obtain a realistic impression of the situation.

Guidance of the rover was achieved in two steps: at the start of a day, the human operator examined the 3D view and planned a route for the rover in terms of a sequence of waypoints through which it would be required to pass. The operator was able to manipulate a graphical representation of the rover and place it at the desired positions to input the information. Then, during the day, the rover proceeded autonomously along the trail of waypoints, monitoring its directional progress with odometric sensors.

While heading for a particular waypoint, frequent stops were made to check for hazards. Obstacles in the path were searched for using two forward-facing CCD cameras and 5 lasers projecting vertical stripes in a diverging pattern. From stereo views of the scene with each of the lasers turned on individually, a triangulation calculation allowed the location of any obstacles to be found. If an obstacle sufficiently obstructed the rover, a local avoiding manoeuvre was planned and executed. If for some reason the rover was unable to return to its original path within a specified time after avoiding an obstacle in this way, the robot stopped to wait for further commands from earth.

So overall, the degree of autonomy in the navigation performed by the rover was actually quite small. The human operator carried out most of the goal selection, obstacle avoidance and path-planning for the robot during the manual waypoint input stage. The separation of waypoints could be quite small in cases where particularly complex terrain was to be traversed. It was certainly necessary to enable the rover to operate autonomously

to the extent described since communication limitations prohibited continuous control from a ground-based operator. However, the rover's low velocity meant that the motions to be carried out between the times when human input was available were small enough so that a large amount of manual assistance was possible. This approach has proved to be very successful in the Pathfinder experiment. The duration of the mission and huge amount of data gathered prove the rover to be a very impressive construction.

1.1.4 This Thesis

The work in this thesis describes progress towards providing a robot with the capability to navigate safely around an environment about which it has little or no prior knowledge, paying particular attention to building and using a useful quantitative map which can be maintained and used for extended periods. The ideas of active vision will be turned to this problem, where visual sensors are applied purposively and selectively to acquire and use data.

In Section 1.2, some of the major previous approaches to robot navigation will be reviewed. Section 1.3 then discusses active vision, and in Section 1.4 we will summarise the approach taken in this thesis to making use of it for autonomous navigation.

The rest of the thesis is composed as follows:

- Chapter 2 is a description of the hardware and software used in the work. The construction of the robot and active head and the computer used to control them will be discussed.
- Chapter 3 introduces the notation and mathematical models of the robot to be used in the following chapters.
- Chapter 4 describes various vision techniques and tools which are part of the navigation system. An initial implementation of these methods in an active version of the 8-point structure from motion algorithm is presented with results.
- Chapter 5 is the main exposition of the map-building and localisation algorithm used. Results from robot experiments with ground-truth comparisons are presented.
- Chapter 6 discusses continuous tracking of features as the robot moves and how this is achieved in the system. A new way to update the robot and map state efficiently is described. A strategy for actively selecting features to track is then developed, and results are given.
- Chapter 7 describes how the work in this thesis is aimed at becoming part of a fully autonomous navigation system. Experiments in goal-directed navigation are presented.
- Chapter 8 concludes the thesis with a general discussion and ideas for future work.

1.2 Major Projects in Robot Navigation

In this section we will take a look at several of the most well-known robot navigation projects, introduce related work and make comparisons with the work in this thesis. The

projects have a wide range of aims and approaches, and represent a good cross-section of current methodology.

1.2.1 INRIA and Structure From Motion

Research into navigation [103, 2, 102] at INRIA in France has been mainly from a very geometrical point of view using passive computer vision as sensory input. Robot navigation was the main initial application for methods which, now developed, have found greater use in the general domain of *structure from motion*: the analysis of image sequences from a moving camera to determine both the motion of the camera and the shape of the environment through which it is passing. Structure from motion is currently one of the most exciting areas of computer vision research.

The key to this method is that the assumption of rigidity (an unchanging scene, which is assumed in the majority of navigation algorithms including the one in this thesis) provides constraints on the way that features move between the images of a sequence. These features, which are points or line segments, are detected in images using interest operators (such as those described in [98, 87, 37, 19]). Initial matching is performed by assuming small camera motions between consecutive image acquisition points in the trajectory, and hence small image motions of the features. If a set of features can be matched between two images (the exact number depending on other assumptions being made about the motion and camera), the motion between the points on the trajectory (in terms of a vector direction of travel and a description of the rotation of the camera) and the 3D positions or *structure* of the matched features can be recovered. An absolute scale for the size of the motion and spacing of the features cannot be determined without additional external information, due to the fundamental depth / scale ambiguity of monocular vision: it is impossible to tell whether the scene is small and close or large and far-away.

The two-view case of structure from motion, where the motion between a pair of camera positions and the structure of the points viewed from both is calculated, has been well studied since its introduction by Longuet-Higgins [56] and Tsai and Huang [97] in the early 1980s. An active implementation of one of the first two-view structure from motion algorithms [99] will be presented in Section 4.3. Recently, this work has been extended, by Faugeras at INRIA and others [29, 39], to apply to the case of uncalibrated cameras, whose internal parameters such as focal length are not known. It is generally possible to recover camera motion and scene structure up to a *projective ambiguity*: there is an unknown warping between the recovered and true geometry. However, it has been shown that many useful things can still be achieved with these results. Much current work in geometric computer vision [60, 1, 90] is tackling the problem of self-calibration of cameras: making certain assumptions about their workings (such as that the intrinsic parameters don't change) allows unwarped motions and scene structure to be recovered.

A more difficult problem is applying structure from motion ideas to long image sequences, such as those obtained from a moving robot over an extended period. Since none of the images will be perfect (having noise which will lead to uncertain determination of feature image positions), careful analysis must be done to produce geometry estimations which correctly reflect the measurement uncertainties. For many applications, it is not necessary to process in real time, and this simplifies things: for instance, when using an image sequence, perhaps obtained from a hand-held video camera, to construct a 3D model of a

room or object, the images can be analysed off-line after the event to produce the model. Not only is it not necessarily important to perform computations quickly, but the whole image sequence from start to end is available in parallel. There have been a number of successful "batch" approaches which tackle this situation [94].

The situation is quite different when the application is robot navigation. Here, as each new image is obtained by the robot, it must be used in a short time to produce estimates of the robot's position and the world structure. Although in theory the robot could use all of the images obtained up to the current time to produce its estimates, the desirable approach is to have a constant-size state representation which is updated in a constant time as each new image arrives — this means that speed of operation is not compromised by being a long way along an image sequence. Certainly the images coming *after* the current one in the sequence are not available to improve the estimation as they are in the batch case.

These requirements have led to the introduction of Kalman Filtering and similar techniques (see Section 5.1) into visual navigation. The Kalman Filter is a way of maintaining estimates of uncertain quantities (in this case the positions of a robot and features) based on noisy measurements (images) of quantities related to them, and fulfills the constant state size and update time requirements above. One of the first, and still one of the most successful, approaches was by Harris with his DROID system [35]. DROID tracked the motion of point "corner" features through image sequences to produce a 3D model of the feature positions and the trajectory of the camera, or robot on which that camera was mounted in a fixed configuration, through the scene. These features are found in large numbers in typical indoor or outdoor scenes, and do not necessarily correspond to the corners of objects, just regions that stand out from their surroundings. A separate Kalman Filter was used to store the 3D position of each known feature with an associated uncertainty. As each new image was acquired, known features were searched for, and if matched the estimates of their positions were improved. From all the features found in a particular image, an estimate of the current camera position could be calculated. When features went out of view of the camera, new ones were acquired and tracked instead. Camera calibration was known, so the geometry produced was not warped. Although DROID did not run quite in real time in its initial implementation (meaning that image sequences had to be analysed after the event at a slower speed), there is no doubt that this would easily be achieved with current hardware. The results from DROID were generally good, although drift in motion estimates occurred over long sequences — consistent errors in the estimated position of the camera relative to a world coordinate frame led to equivalent errors in estimated features positions. This is a common finding with systems which simultaneously build maps and calculate motion, and one which is tackled in this thesis (see Chapter 5).

Beardsley [7, 8, 6] extended the work of Harris by developing a system that used an uncalibrated camera to sequentially produce estimates of camera motion and scene point structure which are warped via an unknown projective transformation relative the ground-truth. Using then approximate camera calibration information this information was transformed into a "Quasi-Euclidean" frame where warping effects are small and it was possible to make use of it for navigation tasks. There are advantages in terms of esthetics and ease of incorporating the method with future self-calibration techniques to applying calibration information at this late stage rather than right at the start as with DROID. In [6], controlled motions of an active head are used to eliminate the need for the input of calibration information altogether: the known motions allow the geometry warping to be reduced to

an affine, or linear, form, and it is shown that this is sufficient for some tasks in navigation such as finding the midpoints of free-space regions.

Returning to work at INRIA, Ayache and colleagues [2] carried out work with binocular and trinocular stereo: their robot was equipped with three cameras in a fixed configuration which could simultaneously acquire images as the robot moved through the world. Their plan was to build feature maps similar to those in DROID, but using line segments as features rather than corners. Having more than one camera simplifies matters: from a single positions of the robot, the 3D structure of the features can immediately be calculated. Using three cameras rather than two increases accuracy and robustness. Building large maps is then a process of fusing the 3D representations generated at each point on the robot's trajectory. Zhang and Faugeras in [103] take the work further, this time using binocular stereo throughout. The general problem of the analysis of binocular sequences is studied. Their work produces good models, but again there is the drift problem observed with DROID. Some use is made of the odometry from the robot to speed up calculations.

It remains to be seen whether structure from motion techniques such as these become part of useful robot navigation systems. It is clearly appealing to use visual data in such a general way as this: no knowledge of the world is required; whatever features are detected can be used. Several of the approaches (e.g. [6]) have suggested using the maps of features detected as the basis for obstacle avoidance: however, point- or line segment- based maps are not necessarily dense enough for this. Many obstacles, particular in indoor environments such as plain walls or doors, do not present a feature-based system with much to go on.

The methods are certainly good for determining the local motion of a robot. However, those described all struggle with long motions, due to the problem of drift mentioned earlier, and their ability to produce maps which are truly useful to a robot in extended periods of navigation is questionable. By example, another similar system is presented by Bouget and Perona in [10]: in their results, processed off-line from a camera moved around a loop in a building's corridors, build-up of errors means that when the camera returns to its actual starting point, this is not recognised as such because the position estimate has become biased due to oversimplification of the propagation of feature uncertainties. The systems do not retain representations of "old" features, and the passive camera mountings mean that a particular feature goes out of view quite quickly as the robot moves. As mentioned earlier, we will look at an active solution to these problems in Chapter 5. Of course, another solution would be to incorporate structure from motion for map-building with a reliable beacon-based localisation system such as a laser in the controlled situations where this is possible. In outdoor, large-scale environments this external input could be GPS.

Some recent work has started to address these problems with structure with motion [96, 95]. One aspect of this work is fitting planes to the 3D feature points and lines acquired. This means that scenes can be represented much better, especially indoors, than with a collection of points and lines. Another aspect is an attempt to correct some of the problems with motion drift: for example, when joining together several parts of a model acquired in a looped camera trajectory, the different pieces have been warped to fit smoothly. While this does not immediately seem to be a rigorous and automatic solution to the problem, it can vastly improve the quality of structure from motion models.

1.2.2 The Oxford Autonomous Guided Vehicle Project

Oxford's AGV project [18] has been an attempt to draw together different aspects of robot navigation, using various sensors and methods, to produce autonomous robots able to perform tasks in mainly industrial applications. One of the key aspects has been sensor fusion: the combining of output from different sorts of devices (sonar, vision, lasers, etc.) to overcome their individual shortcomings.

Some of the most successful work, and also that which has perhaps the closest relationship to that in this thesis, has been that of Durrant-Whyte and colleagues on map-building and localisation using sonar [52, 53, 54, 57, 27]. They have developed tracking sonars, which actively follow the relative motion of world features as the robot moves. These are cheap devices of which several can be mounted on a single robot. The sonar sensors return depth and direction measurement measurements to objects in the scene and are used for obstacle avoidance, but they are also used to track landmarks features and perform localisation and map-building. The "features" which are tracked are regions of constant depth (RCD's) in the sonar signal, and correspond to object corners or similar.

Much of the work has been aimed at producing localisation output in known surroundings: the system is provided with an a priori map and evaluates its position relative to it, and this has been shown to work well. This problem has much in common with model-based pose estimation methods in computer vision [35, 40], where the problem is simply one of finding a known object in an image.

Attention has also been directed towards map-building and localisation in unknown surroundings. In his thesis, Leonard [52] refers to the work at INRIA discussed previously as map fusing — the emphasis is on map *building* rather than map *using*. This is a very good point: the dense maps of features produced by structure from motion are not necessarily what is needed for robot navigation. In the sonar work, landmark maps are much sparser but more attention is paid to how they are constructed and used, and that is something that the work in this thesis addresses, now with active vision. Lee and Recce [51] have done work on evaluating the usefulness of maps automatically generated by a robot with sonar: once a map has been formed, paths planned between randomly selected points are tested for their safety and efficiency.

Another comment from Leonard is that when choosing how a robot should move, sometimes retaining map contact is more important than following the shortest or fastest route. The comparison is made with the sport of orienteering, where runners are required to get from one point to another across various terrain as quickly as possible: it is often better to follow a course where there are a lot a landmarks to minimise the chance of getting lost, even if this distance is longer. This behaviour has been observed in animals as well [69], such as bees which will follow a longer path than necessary if an easily recognised landmark such as a large tree is passed.

It should be noted that sonar presents a larger problem of *data association* than vision: the much smaller amount of data gathered makes it much more difficult to be certain that an observed feature is a particular one of interest. The algorithm used for sonar map-building in this work will be looked at more closely in Chapter 5. Durrant-Whyte is now continuing work on navigation, mainly from maps, with applications such as underwater robots and mining robots, at the University of Sydney [82, 81].

Other parts of the AGV project have been a laser-beacon system for accurate localisation

in a controlled environment, which works very well, and a parallel implementation of Harris' DROID system [34] providing passive vision. Work has also taken place on providing the AGV with an active vision system, and we will look at this in Section 1.4. The problem of path-planning in an area which has been mapped and where a task has been defined has also received a lot of attention. Methods such as potential fields have a lot in common with path planning for robot arms. However, there has not yet been much need for these methods in mobile robotics, since the more pressing and difficult problems of localisation, map-building and obstacle detection have yet to be satisfactorily solved. It seems that a comparison can be drawn with other areas of artificial intelligence — computers are now able to beat the best human players at chess, where the game is easily abstracted to their representations and strengths, but cannot perform visual tasks nearly as well as a baby. Perhaps if this abstraction can one day be performed with real-world sensing, tackling path-planning will be straightforward.

1.2.3 MIT: Intelligence Without Representation

A quite different approach to navigation and robotics in general has been taken by Brooks and his colleagues at MIT [12, 13, 14]. Their approach can be summarised as "Intelligence Without Representation" — their robot control systems consist of collections of simple behaviours interacting with each other, with no central processor making overall decisions or trying to get an overall picture of what is going on. The simple behaviours themselves work with a minimum of internal representation of the external world — many are simple direct reflex-type reactions which couple the input from a sensor to an action.

A core part of their ideology is that the secrets of designing intelligent robots are only uncovered by actually making and testing them in the real world. Working robot systems are constructed in layers, from the bottom upwards: the most basic and important competences required by the robot are implemented first, and thoroughly tested. Further levels of behaviour can then be added sequentially. Each higher level gets the chance to work when control is not being demanded from one of the ones below, this being referred to as a subsumption architecture. For a mobile robot designed to explore new areas, the most basic level might be the ability to avoid collisions with objects, either stationary or moving, and could be implemented as tight control loops from sonar sensors to the the robot's drive motors. The next layer up could be a desire to wander the world randomly — if the robot is safe from obstacles, it will choose a random direction to drive in. Above this could be a desire to explore new areas: if random wandering revealed a large unfamiliar region, this behaviour would send the robot in that direction.

The result is robots which seem to behave in almost "life-like" ways. Patterns of behaviour are seen to emerge which have not been directly programmed — they are a consequence of the interaction of the different control components. Some of the most impressive results have been seen with legged insect-like robots, whose initially uncoupled legs learn to work together in a coordinated fashion to allow the robot to walk forwards (via simple feedback-based learning).

Brooks' current work is largely aimed towards the ambitious aim of building an artificial human-like robot called Cog [15]. Much of the methodology developed for mobile robots has been maintained, including the central belief that it is necessary to build a life-size human robot to gain insights into how the human brain works. However, it seems that much of

the MIT group's enthusiasm for mobile robots has faded without actually producing many robots which can perform useful tasks. While there is no doubt that the work is fascinating in terms of the parallels which can be drawn with biological systems, perhaps a slightly different approach with more centralised control, indeed using computers to do the things they can do best like dealing accurately and efficiently with very large amounts of complex data, will be more fruitful. Certain aspects of the methodology, such as the layered hierarchy of skills, do seem to be exactly what a robust robot system will require.

1.2.4 Carnegie Mellon University: NavLab

The large NavLab project started at CMU in the early 1980's has had two main branches: the Unmanned Ground Vehicles (UGV) group, concerned with the autonomous operation of off-road vehicles, and the Automated Highways Systems (AHS) group, working to produce intelligent road cars (and also intelligent roads) which reduce or eliminate the work that human drivers must do.

The UGV group [50, 47] have have taken an approach using a variety of sensors, and applied it to the difficult problem of off-road navigation, with full-size vehicles as testbeds. The sensors used include vision and laser range scanners. A key part of the work is developing algorithms for terrain classification, and then building local grid-based maps of the areas which are safe to drive through. Planning of vehicle movements takes place via a voting and arbiter scheme, having some common ground with the MIT methodology. Different system sensing and behaviour modules vote towards the desired action, and are weighted according to their importance — safety-critical behaviours such as obstacle avoidance have the highest weights. The system has shown good performance, with autonomous runs of up to a kilometer over unknown territory reported. Global map-building is not a current part of the system, but could potentially be incorporated with the help of GPS for example.

The AHS branch [72, 46, 5] has been concerned with enabling road cars to control their own movements, relieving bored and dangerous human drivers. On-road navigation is a much more highly-constrained problem than off-road: certain things can usually be counted on, such as the clearly identifiable lines running along the centre of a road, or a rear view of the car in front. Techniques developed have allowed cars to follow the twists of a road, keep safe distances from other vehicles, and even perform more complicated manoeuvres such as lane-changes. Neural networks have often been employed, and sensors have included vision and radar. However, the huge distances covered every day by cars mean that such systems would have to be incredibly reliable to be safer than human drivers. The most ambitious experiment attempted by the AHS group was called "No Hands Across America" in 1995, where a semi-autonomous car was driven from coast-to-coast for nearly 3000 miles across the USA. During the journey, the researchers on board were able to take control from the system whenever it became necessary. In the course of the experiment, the car controlled its own steering for 98.2% of the time (accelerator and brake were always human-controlled). This figure is highly impressive, considering the wide variety of roads and conditions encountered on the trip. However, it still means that for about 50 miles the researchers had to take control, and these miles were composed of many short intervals where conditions became too tricky for the computer. It was necessary for the human driver to remain alert at nearly all times.

The constrained nature and high speed of on-road driving makes it seem a quite differ-

ent problem to multi-directional navigation around an area. In human experience, driving through somewhere does not necessarily provide much information which would be useful if the place was later revisited on foot. Knowledge of the vehicle's world position is not important in road driving. Something that makes this clear is the fact that early driving simulators and games did not produce in-car viewpoints from internally stored 3D representations of the shape of a road or track (as modern ones do), but generated fairly realistic graphics from simpler constructs such as "show a tight right-hand bend for 5 seconds".

1.3 Active Vision

Although computer vision has until now not been the most successful sensing modality used in mobile robotics (sonar and infra-red sensors for example being preferred), it would seem to be the most promising one for the long term. While international research into computer vision is currently growing rapidly, attention is moving away from its use in robotics towards a multitude of other applications, from face recognition and surveillance systems for security purposes to the automatic acquisition of 3D models for Virtual Reality displays. This is due to the greater demand served by these applications and the successes being enjoyed in their implementation — they are certainly the immediate future of computer vision, and some hugely impressive systems have been developed. However, we feel that it is well worth continuing with work on the long-term problems of making robot vision systems.

Vision is the sensor which is able to give the information "what" and "where" most completely for the objects a robot is likely to encounter. Although we must be somewhat cautious of continuous comparisons between robot and biological systems [69], it is clear that it is the main aid to navigation for many animals.

Humans are most certainly in possession of an *active vision* system. This means that we are able to concentrate on particular regions of interest in a scene, by movements of the eyes and head or just by shifting attention to different parts of the images we see. What advantages does this offer over the *passive* situation where visual sensors are fixed and all parts of images are equally inspected?

- Parts of a scene perhaps not accessible to a single realistically simple sensor are viewable by a moving device — in humans, movable eyes and head give us almost a full panoramic range of view.
- By directing attention specifically to small regions which are important at various times we can avoid wasting effort trying always to understand the whole surroundings, and devote as much as possible to the significant part — for example, when attempting to perform a difficult task such as catching something, a human would concentrate solely on the moving object and it would be common experience to become slightly disoriented during the process.

Active vision can be thought of as a more *task* driven approach than passive vision. With a particular goal in mind for a robot system, an active sensor is able to select from the available information only that which is directly relevant to a solution, whereas a passive system processes all of the data to construct a global picture before making decisions — in this sense it can be described as *data* driven.

The emerging view of human vision as a "bag of tricks" [74] — a collection of highly specialised pieces of "software" running on specialised "hardware" to achieve vision goals — rather than a single general process, seems to fit in with active vision ideas if a similar approach is adopted in artificial vision systems. High-level decisions about which parts of a scene to direct sensors towards and focus attention on can be combined with decisions about which algorithms or even which of several available processing resources to apply to a certain task. The flexibility of active systems allows them to have multiple capabilities of varying types which can be applied in different circumstances.

1.3.1 Previous Work in Active Vision

The field of computer vision, now very wide in scope with many applications outside of robotics, began as a tool for analysing and producing useful information from single images obtained from stationary cameras. One of the most influential early researchers was Marr [58], whose ideas on methodology in computer vision became widely adopted. He proposed a data driven or "bottom up" approach to problems whereby all the information obtained from an image was successively refined, from initial intensity values though intermediate symbolic representations to a final 3-dimensional notion of the structure of a scene. While this proved to be very useful indeed in many cases, the process is computationally expensive and not suited to several applications. Clearly the alternative active vision paradigm could be advantageous in situations where fast responses and versatile sensing are required — certainly in many robotics applications. Crucially, in Marr's approach, the task in hand did not affect the processing carried out by the vision system, whereas with the active paradigm resources are transferred to where they are required.

The first propositions of the advantages of an active approach to computer vision were in the late 1980s [3], showing the real youth of this field of research. Theoretical studies concentrated on how the use of a movable camera could simplify familiar vision algorithms such as shape-from-shading or shape-from-contours by easily providing multiple views of a scene. Perhaps, though, this was missing the main advantages to emerge later from active vision, some of which were first pointed out in work published by researchers at the University of Rochester, New York [4, 16]. One of these was the concept of *making use of the world around us as its own best memory.* With a steerable camera, it is not necessary to store information about everything that has been "seen", since it is relatively simple to go back and look at it again. This is certainly something that occurs in the human vision system: we would be unable to perform many simple tasks involving objects only recently looked at without taking another glance to refresh knowledge of them.

Rochester and others continued the train of research by investigating potential control methods for active cameras. In the early 1990s several working systems began to appear: Rochester implemented their ideas using a stereo robot head based on a PUMA industrial robot (to be followed later by a custom head), and Harvard University developed a custom stereo head [31]. Another advanced stereo platform was built by Eklundh's group in Sweden [70].

Around this time the Oxford Active Vision Laboratory was formed and began to work on similar systems. This work will be described in more detail in the next section, as it provides a good representation of current active vision research in general.

(a) (b) (c)

Figure 1.1: The Yorick series of active robot heads: (a) the original model used in surveillance, (b) Yorick 8-11 which is mounted on a vehicle and used for navigation, and (c) Yorick 55C which is small enough to be mounted on a robot arm and used in telepresence applications.

The Oxford Active Vision Project

Active vision research in Oxford has taken a mechatronic approach to providing the directability of visual attention required by the paradigm. Early work [20] made use of cameras mounted on robot arms, but it soon became clear that more specialised hardware was required to produce the high performance camera motions required by useful vision algorithms, and this led to the construction on the Yorick series of active stereo heads (see Figure 1.1). Designed by Paul Sharkey, there are currently three, each aimed at a certain application and differing primarily in size, but generically similar in their layout — all have two cameras and four axes of rotation (as in Figure 2.5).

The first Yorick [84] is a long-baseline model (about 65cm inter-ocular spacing) designed to be used for the dynamic surveillance of scenes. In surveillance, a robot observer must be able to detect, switch attention to and follow a moving target in a scene. By splitting images obtained from an active camera into a low resolution periphery and a high resolution central area or *fovea*, this can be achieved in a way which is potentially similar to how animal and human vision systems operate: any motion in the scene can be detected by measuring optical flow in the coarse outer region, and then this information can be used to fire a *saccade* or rapid redirection of gaze direction so that the camera is facing directly towards the object of interest [67]. The saccades are programmed so that when the camera arrives at a position to look directly at the moving object, it is also moving at the estimated angular velocity necessary to follow the object if it continues to move at the same speed. This makes it fairly simple to initiate some kind of tracking of the motion since the head will follow it for a short while anyway if the acceleration of the object is not too large [11].

Tracking of objects has been performed in two ways. The aim has been to keep the head moving such that the object of interest is continuously in the central foveal region of the image. The high definition of this area provides the detail necessary to detect small deviations from perfect alignment when corrections are needed. Initially, optical flow methods were used in the fovea to successfully track objects over extended periods. While this worked well, the very general nature of the method meant that it could break down — any motion in the fovea was simply grouped together and the average of the optical flow vectors

for all pixels were averaged to find a mean velocity of the object. If the object passed close to other moving parts of the scene, this could be problematic since the assumption that there was only one moving body in the fovea would no longer hold. Also, the fixation point of the camera on the object was not very well defined since just the centre of the detected moving "blob" was used — rotations would mean that different parts of the object were fixated. One advantage of this was a flexibility allowing deformable objects such as people to be tracked.

Later tracking procedures used point features or "corners" on objects. These well located points could be matched between successive frames of a tracking sequence to reveal how certain parts of the object were moving. In order to determine a fixation point to use to track the whole object the method of affine transfer was developed [76], since it was discovered that just using one of the points or a mean of all of those found in a certain frame was unstable (a slightly different set of points would be visible in each frame). This is a way of calculating a stable fixation point on an object from whatever corners are detected in each frame. Since corner features can also be matched between images obtained from two cameras, this is also one of the pieces of work so far which has been fully implemented in stereo [28].

The large size of the first Yorick was chosen to give good performance when recovering the depth of scene features using stereo in the kind of scenes likely to be encountered (see Section 4.2.2 for a discussion of this). Geared (as opposed to direct-drive) motors were used throughout, as it was determined that this would be necessary to achieve very high accelerations for the rotation axes due to the relatively large moment of inertia of the head. High acceleration performance was one of the primary design requirements of the head since it is this characteristic which would allow it to replicate the rapid response of the human eyes and head when looking at new features or closely tracking objects. The large head has also been used to recover the structure of stationary parts of scenes [77] in work towards the automatic active exploration of scene geometry, and also in work on the automatic determination of camera calibration parameters [64].

With attention turning towards the use of active vision for robot navigation, an active head with similar high performance but small enough in size to be mounted on a vehicle was required. Yorick 8-11, depicted in Figure 1.1(b), has a baseline of about 34cm. Active navigation has used the high saccade speed of this active head to fixate rapidly on different landmarks [24], and its close-control capabilities to continuously track features to aid steering [6, 68], both in prelude to the work presented in this thesis. Other active navigation work, concerned with navigation in tight clearances [55] and automated stereo calibration, has proceeded with a lower performance head.

The third head (Figure 1.1(c)) is much smaller than the others, with camera spacing comparable to the human inter-ocular distance. It is small enough to be mounted on a robot arm. Work using this head has been aimed towards telepresence — the idea that it can act as someone's eyes in a remote location: perhaps somewhere too distant or dangerous for a human to go. Using a visual head tracker to follow the head of a person wearing a head-mounted display, the Yorick head has been slaved to mimic the person's movements and feed back what it is seeing to the HMD [41, 40]. While not really an active vision application in this form, it is hoped to extend this work using vision algorithms to control the head more autonomously.

1.4 This Thesis: Navigation Using Active Vision

There have so far only been limited efforts to apply active vision to navigation, perhaps largely because of the complexity of building mobile robots with active heads, which only a few laboratories are in a position to do.

1.4.1 Some Previous Work in Active Visual Navigation

Li [55] built on the work of Du [26], in using an active head to detect obstacles on the ground plane in front of a mobile robot, and also performing some active tracking of features with a view to obstacle avoidance. However, only limited results were presented, and in particular the robot did not manoeuvre in response to the tracking information, and no attempt was made to link the obstacle-detection and obstacle-avoidance modules.

Beardsley *et al.* [6] used precisely controlled motions of an active head to determine the affine structure of a dense point set of features in front of a robot, and then showed that it was possible to initiate obstacle avoidance behaviours based on the limits of free-space regions detected. This work was impressive, particularly as it operated without needing knowledge of camera calibrations. It implemented in short demonstrations which worked well, but consideration was not given to the long-term goals of navigation and how the active head could be used in many aspects.

At NASA, Huber and Kortenkamp [42, 43] implemented a behaviour-based active vision approach which enabled a robot to track and follow a moving target (usually a person). Correlation-based tracking was triggered after a period of active searching for movement, and extra vision modules helped the tracking to be stable. The system was not truly an active vision navigation system, however, since sonar sensors were used for obstacle detection and path planning.

1.4.2 A Unified Approach to Navigation with Active Vision

Active cameras potentially provide a navigating vehicle with the ability to fixate and track features over extended periods of time, and wide fields of view. While it is relatively straightforward to apply fixating vision to tactical, short-term navigation tasks such as servoing around obstacles where the fixation point does not change [68], the idea of using serial fixation on a succession of features to provide information for strategic navigation is more involved. However, active vision is seemingly well-suited to this task: the ability to measure features over such a wide range means that the same ones can be used as a robot makes a wide range of movements. This has advantages for map-building and localisation over the use of passive cameras with a narrow field of view.

The core work of this thesis concerns the problem of simultaneous localisation and map-building for a robot with a stereo active head (Chapter 5), operating in an unknown environment and using point features in the world as visual landmarks (Chapter 4). Importance has been attached to producing maps which are useful for *extended* periods of navigation. As discussed in Section 1.2, many map-building methods fail on extended runs because they do not have the ability to recognise previously visited areas as such and adjust their maps accordingly. With active cameras, the wide field of view means that we really can expect to re-detect features found a long time ago, even if the area is not passed through along the original trajectory. Maintaining a large, consistent map requires detailed

information to be stored and updated about the estimates of all the features and their relationships. This information is computationally expensive to maintain, but a sparse map of landmark features can be handled successfully. In Section 6.2, a method is presented which dramatically increases the efficiency of updates in the case that repeated measurements are made of a single feature. This permits continuous real-time tracking of features to occur irrespective of the total map size.

The selective approach of active sensing means that a strategy must be developed for serially fixating on different features. This strategy is based on decisions about which measurements will provide the best information for maintaining map integrity, and is presented in Section 6.3. When combined with criteria for automatic map maintenance in Chapter 7 (deciding when to add or delete features), a fully automatic map-building system is achieved.

Active sensing, when applied to navigation, could mean more than just selectively applying the sensors available: the path to be followed by the robot could be altered in such a way as to affect the measurements collected. This approach is used by Blake *et al.* [9], where a camera mounted on a robot arm makes exploratory motions to determine the occluding contours of curved obstacles before moving in safe paths around them. Whaite and Ferrie [100] move an active camera in such a way as to maximise the information gained from a new viewpoint when determining the shape of an unknown object. In this thesis, it is assumed that the overall control of the robot's motion comes from a higher-level process, such as human commands, or behaviours pulling it to explore or reach certain targets. The navigation capabilities sought should act as a "server" to these processes, supporting them by providing the necessary localisation information.

We will discuss in detail in Chapters 5 and 7 the tasks that an autonomous robot should be able to perform. In most applications, there will be at least some prior information or commands governing the required motion and we will look at how this information can be incorporated with map-building techniques designed for unknown environments. We will make the distinction between position-based navigation, and so-called context-based navigation, where the robot manoeuvres with respect to locally observed parts of the surroundings. A simple steering law is used to direct the robot sequentially around obstacles or through waypoints on a planned trajectory.

Throughout the thesis, a variety of experiments will be presented which test the capabilities developed in a realistic environment and make extensive comparisons between estimated quantities and ground-truth. The system has been fully implemented on a real robot, incorporating such capabilities as continuous fixated feature tracking and dynamic steering control, and we will discuss system details as well as the mathematical modelling of the robot system necessary for accurate performance.

Some work using sonar rather than vision, but with an active approach which has perhaps most in common with this thesis was by Manyika [57]. Although sonar presents a different set of problems to vision, especially in terms of the more difficult feature-correspondence issues which must be addressed, the use of tracking sensors to selectively make continuous measurements of features is closely related. He was concerned with sensor management: applying the available resources to the best benefit. However, the robot in Manyika's work operated in known environments, recognising features from a prior map, greatly simplifying the filtering algorithm used.

2

The Robot System

The aim of this thesis, novel research into robot navigation using active vision, has demanded an experimental system with the capability to support the modes of operation proposed. Since the aim of this thesis has been vision and navigation research rather than robot design, efforts have been made to keep the system simple, general (in the sense that apart from the unique active head, widely available and generally functional components have been preferred over specialised ones), and easy to manage. This characterises the recent move away from highly specialised and expensive hardware in computer vision and robotics applications, and more generally in the world of computing where the same computer hardware is now commonly used in laboratories, offices and homes.

2.1 System Requirements

The system requirements can be broken down into four main areas:

- Robot vehicle platform
- Vision acquisition and processing
- Active control of vision
- System coordination and processing

The approach to navigation in this thesis is suited to centralised control of the system components. From visual data acquired from the robot's cameras, decisions must be made about the future behaviour of the cameras and the motion of the vehicle. Signals must then be relayed back to the active camera platform and vehicle. Their motion will affect the

Figure 2.1: The robot used in this thesis comprises an active binocular head, equipped with two video cameras, mounted on a three-wheeled vehicle.

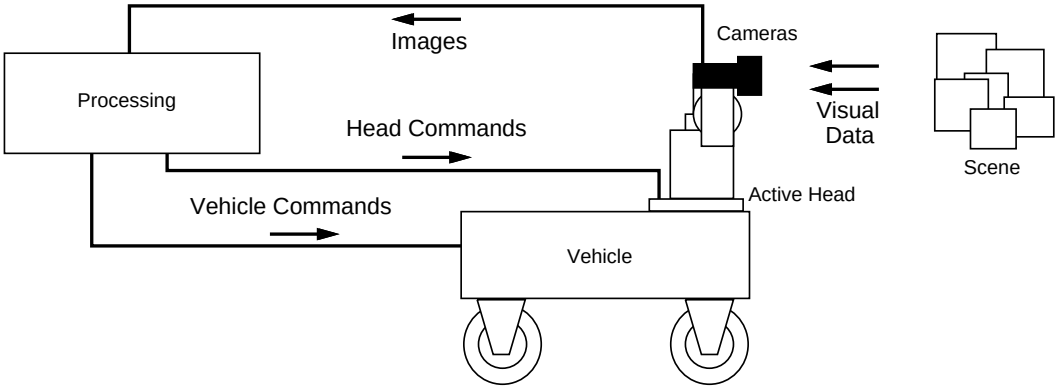


Figure 2.2: The basic flow of information around the robot system.

next batch of visual data acquired. The essential flow of information around the system is depicted in block form in Figure 2.2.

2.1.1 System Composition

There have been two stages in putting the system together in its current form. The first, which was in place when the work in this thesis was started, made use of the hardware available at the time from other projects to make a working, but convoluted system. In particular, the setup included a Sun Sparc 2 workstation acting as host and an i486 PC hosting the head controller and transputers communicating with the vehicle. Image acquisition was performed by external Datacube MaxVideo boards, and most of the image processing was carried out on a network of transputers, again separately boxed from the rest of the system.

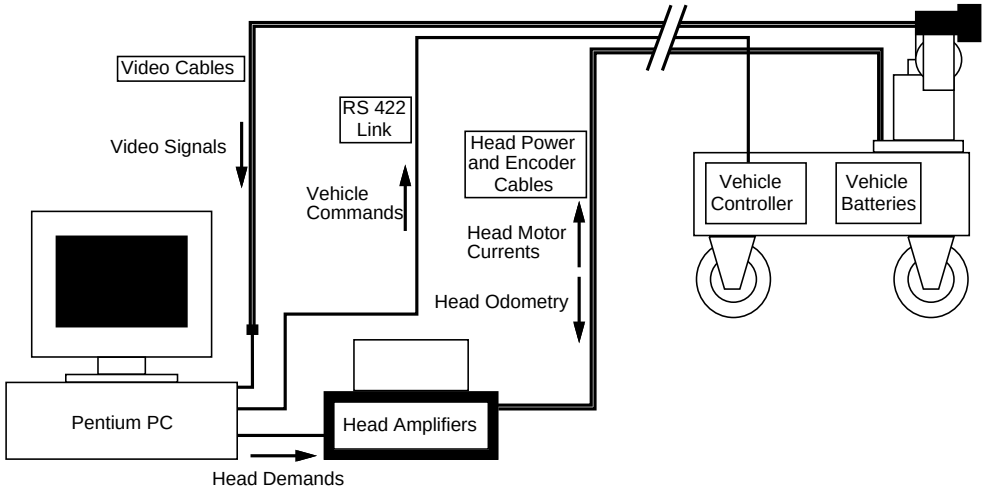


Figure 2.3: System configuration: the robot is connected to its base-station, consisting of a Pentium PC and externally boxed head amplifiers, by an RS422 link (vehicle control), video cables (image acquisition), and motor and encoder cables (head control).

It was decided to rearrange the system significantly with the primary aim of reducing its complexity (with portability in mind so that the system could be moved to different locations for experiments). Its current configuration is shown in block form in Figure 2.3. It can be seen that a Pentium PC now serves as host, and that as many components as possible have actually been located inside this PC. The PC and head motor amplifiers are now the only units external to the robot itself. An attempt was not made to make the robot completely self-contained in the sense that it should carry everything on-board. While this would have had the obvious benefit of making the robot completely free-roaming and unhindered in its range or variety of manoeuvres by cables, it was decided that it would have led to an unacceptable increase in system complexity, requiring all components to be small and light enough to be carried by the robot and also to be powered by its on-board batteries.

In the rest of this chapter, the individual components of the robot system will be looked at in more detail.

2.2 A PC/Linux Robot Vision System

Basing the system around a Pentium PC has succeeded in bringing most of it together into one box, and has proven to be an inexpensive and flexible way to run a robot. The PC uses Linux, the public domain UNIX operating system which was chosen primarily because of its compatibility with other machines on the laboratory's local network for ease of development.

2.2.1 PC Hardware

The PC has an Intel Pentium P100 processor (which is easily upgradable to a faster model should the need arise) and 32 Megabytes of RAM. Modern PC-compatible computers rep-

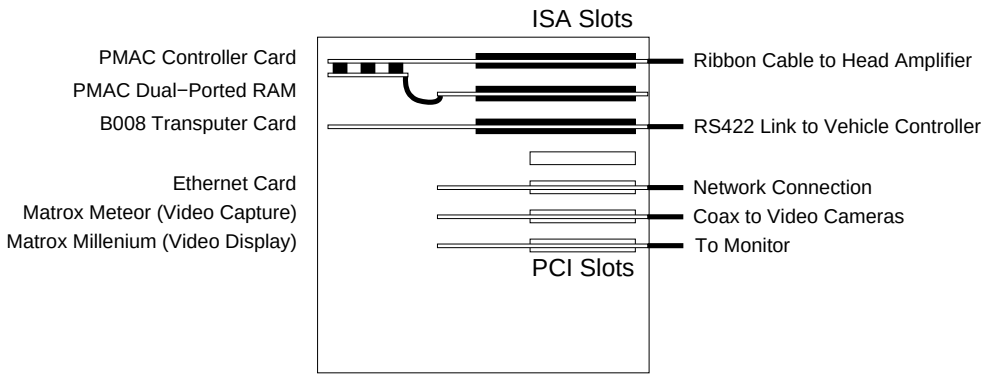


Figure 2.4: The interior of the host PC: its three ISA bus expansion slots are filled with a PMAC controller card, the PMAC's dual-ported memory card, and a B008 transputer card fitted with one transputer processor. Of its four PCI bus slots, three are filled with an ethernet card, a Matrox Meteor video capture card and a Matrox Millennium video display card.

resent good value in terms of processing power compared to alternative UNIX workstations, although these often provide superior graphics and other features which are useful in many vision applications.

A major reason for the choice of a PC over other possibilities is the wide range of expansion cards available which can be slotted simply into the computer's motherboard. Figure 2.4 shows the way the PC's expansion slots have been used. The older cards (PMAC and B008) surviving from the system's previous incarnation are fitted into the ISA bus expansion ports, while the newer cards purchased specially for the new system are attached to the ports on the superior PCI (Peripheral Component Interface) bus.

The cards installed in the PC are:

- Delta Tau PMAC motor controller card and dual-ported RAM card (head control).
- Inmos B008 transputer card (vehicle communications).
- Matrox Meteor video capture card.
- Matrox Millennium video display card.
- SMC Etherpower network card.

2.2.2 Operating System

The increasing use of Linux has led to a continual improvement in the level of support available for different devices and modes of operation. As mentioned above, the primary reason for its choice in the system was the compatibility it provided with the many UNIX machines already in use in the laboratory. The PC functions as a general workstation when the robot is not in use.

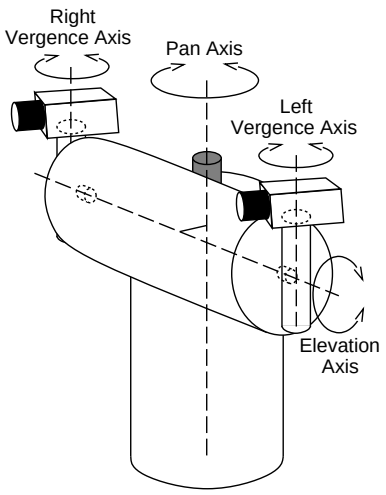


Figure 2.5: Yorick 8-11 has pan, elevation and twin vergence axes of movement.

Linux is an operating system which lies entirely in the public domain, meaning that while it comes with no guarantees or software support, the user is free to use it for any purpose, and the source code is available so that custom modifications can be made.

Two disadvantages of using Linux were:

- At the time of the reconfiguration of the system, Linux device drivers were not available for either the Matrox Meteor video capture card or the PMAC motor controller card. It was necessary to confer with others wanting to use these cards under Linux and to play a part in successfully developing the drivers.
- Linux is not a "real-time" operating system in the sense that the timing of programs can be guaranteed; they may be interrupted to perform other system tasks. In practice, this did not prove to be a large problem. The necessary timing signals for the coordination of the composite robot system were obtained from component sources such as the video capture card or the vehicle controller, and allowances made for a potential loss of processing time due a short interruption.

A Linux system can be easily run in "single machine" mode, so that the computer can be disconnected from its local network. This was important so that the system could be portable.

2.3 The Active Head: Yorick

The active camera platform used on the robot, Yorick 8-11, is one of the series of heads built in Oxford already described in Chapter 1.

The purpose of the head is to enable the two video cameras it carries to be directed as required by a robot performing navigation tasks. The "common elevation" head geometry, with four axes of movement as shown in Figure 2.5, means that the two cameras move together as the pan and elevation angles are changed, and can be converged (symmetrically

Description	Vergence	Elevation	Pan
Axis Range	178.5°	177.5°	335°
Max Slew Rate	540°/s	470°/s	430°/s
Max Acceleration	38000°/s^2	25000°/s^2	20000°/s^2
Backlash	0.0075°	0.0075°	0.0025°/s
Angle Resolution	0.00036°	0.00036°	0.00018°
Repeatability(min)	0.00036°	0.00036°	0.00018°
Repeatability(max)	0.005°	0.005°	0.005°
Mass	8.5kg (excluding cameras)		
Payload	2 × 0.5kg		
Baseline	0.338m		

Table 2.1: Performance and specifications of the Yorick 8-11 active head.

or asymmetrically) as the vergence angles change to both point towards the same feature at some depth in the scene. The performance of Yorick 8-11 is detailed in Table 2.1.

To control the head motors, a commercial PMAC controller card manufactured by Delta Tau is used. This card is mounted in the host PC with which it communicates over the ISA bus. A small computer in its own right, with a processor and memory, the PMAC is loaded with a motion program which controls the position of each of the head axes in a tight loop by reading the odometry from the encoders and providing signals to the motors at a very high frequency. Alongside the PMAC card in the PC is mounted a dual-ported RAM module, which is memory that can be accessed directly by either the PMAC or the PC. Commands to the head (such as a desired position for a certain axis) are issued from the PC by writing them into this shared memory. They are then read and interpreted by the PMAC motion program, and the appropriate changes made to the signals sent to the motors in the tight control loop.

2.4 The Vehicle: GTI

The vehicle [17] is a simple but sturdy 3-wheeled model developed in Oxford with a modular design which gave it the flexibility to be used in various robotic projects before it was united with the Yorick head. It has two free-running forward-facing wheels at the front, and a single driveable and steerable wheel at the rear. The steering axis has a full 180° range of movement, and with the steering angle set to ±90° the vehicle can rotate on the spot about the central point on the fixed front axis. (Note that this capability decided the choice of mount-point for the active head — centrally over the fixed axis — and therefore, the choice of the "front" and "back" of the vehicle. In this configuration it is possible to make vehicle movements which result in pure rotations of the active head.) The kinematics of the vehicle are investigated in Chapter 3. It carries its own power supply in the form of four 12V lead acid batteries.

Control of the vehicle's two motors (drive and steering) is achieved using the dedicated OXNAV [88] dual motor control board mounted on the vehicle. Commands are sent to this board via an RS422 link which is connected to an interface on the Inmos B008 transputer card mounted on the ISA bus of the host PC. The B008 hosts just one transputer processor,

whose job is to receive motion commands from the PC and translate them into commands to send to the OXNAV board along the RS422 link.

The control performance of the vehicle is relatively low (certainly compared to the active head), in the sense that a command for a certain motion can produce a result that differs from the command by an unpredictable amount within a reasonably large range. This is due mainly to slippage between the wheels and the floor, but also to the deliberately fairly approximate calibration of the vehicle. However, this is not a problem with regard to the work presented in this thesis, since it is exactly these kind of errors in motion estimation whose effect it is hoped to determine and correct by visual localisation and navigation.

The peak forward velocity of the vehicle is of the order of 20cms^{-1}.

2.5 Image Capture

The cameras used on Yorick 8-11 are compact CCD devices producing standard monochrome PAL output at a frequency of 50Hz. In the robot system, the cameras are powered from the vehicle's 12V batteries.

The Matrox Meteor image capture card is able to grab PAL frames at the full interlaced resolution of 768×576 pixels and make them available to the host PC over the PCI bus at field rate. The Meteor is designed for colour image acquisition from either a composite video source or separate synchronous red, green and blue (RGB) inputs. In the robot system, colour is not required, but there are two monochrome cameras. Simultaneous capture from these was achieved with just one Meteor by connecting the cameras to two of the RGB channels as if their signals were just different colour fields from a single camera. The Meteor composites these signals into a single mixed image where bytes from the left and right camera images alternate in a regular pattern; it is an easy task to separate out the images for processing and displaying.

For speed, in the final system images are subsampled down to 192×144 pixels before processing.

2.6 Software

The approach to software in the system has been simplified by its centralised PC-based construction, which minimised worries about communication between different components. As mentioned earlier, commands to the robot and active head are communicated via their Linux drivers. Images from the framegrabber are read with calls to its driver. A central program running on the PC is therefore able to control all parts of the system. All image and navigation processing are performed by the Pentium processor.

The main program controlling the system was written in C++. An important requirement during the development of the system was the capability to test ideas and algorithms in simulation before full robot implementation. With this in mind, the main program was designed with dual simulation / robot modes of operation as a fundamental feature, and easy switching between the two while keeping as much code as possible in common was permitted. The object-oriented nature of C++ made this easy to achieve. Interchangeable "Robot" and "Simulation" classes were created which had the same interface: a call to a member function of the Robot class would cause an actual result, such as a robot motion,

while the equivalent call to the Simulation class would cause the simulation of the robot and world to change accordingly. A choice at compile-time indicates which mode the program will operate in. The main classes of the program which are common to Simulation and Robot modes are "Scene", which contains and maintains estimates of the positions of the robot and features in world, and "Kalman", which contains functions to perform filter updates. These classes should make it straightforward to transplant the map-building and localisation algorithm of this thesis to a different robot.

For display purposes and the fast processing of matrices, routines from the Horatio [62] vision libraries were used. The need for fast operation dissuaded the use of a dedicated C++ library such as the Image Understanding Environment since at the time of writing work on real-time sections is still in progress.

2.7 Working in Real Time

When moving and tracking features simultaneously, the system executes a loop at a frequency of 5Hz, at each step of which a motion prediction is made, the head moves, images are acquired and displayed, image processing occurs and updates to all the estimated quantities take place. Implementation details will be discussed further in Chapters 5 and 6. However, we will quickly note that the main factor limiting speed of operation was the processing time needed to perform image processing on the stereo images. Vehicle odometry is used to provide the trigger for the loop timing: a new image is acquired when the vehicle has reached a certain position according to readings from its wheel encoder. In this sense, the update frequency is linked to distances travelled by the vehicle, and so 5Hz is only an approximate value calculated from the vehicle calibration. The program has the ability to adjust the rate of measurements automatically in reponse to time over-runs in the processing stages: as will be explained later, this can happen in normal circumstances when there is a large amount of image processing to do, but another possibility is lost processing due to an interruption from another Linux process.

2.8 Calibration

Most aspects of the robot system were used in a calibrated configuration in the work in this thesis. However, the calibrations used were approximate apart from with the active head, whose geometry is very well known.

- Cameras: The focal lengths were determined to two significant figures with a simple measurement of the image size of a known object at a known distance. The principal points used were usually the camera image centres, which have been shown in previous accurate calibrations of the cameras to be close to the true locations. Measurements made by the active head at fixation (see Section 4.2) are much more sensitive to errors in the principal points than the focal lengths. The same calibration values were used for both of the robot's cameras (see Section 3.2 for details of the pinhole camera model).

  Automated camera calibration is now an important area of computer vision research, and self-calibration of stereo rigs has also been tackled [25, 55]. However, it is felt

that this work is somewhat adjacent to the work in this thesis, where the emphasis is on using a calibrated active camera system effectively for navigation.

- Head: Section 3.3 describes the mathematical model used of the active head. The constant dimensions of the head encoded in the constants H, I, p, c and n are well known from the head design or measurements. Yorick 8-11 does not have well defined "end stops" on its axes, or another automatic way to repeatably find their zero locations, so this was normally achieved by moving them by hand each time the system was started. It is estimated that a zero accuracy of around 1° could easily be achieved in this way. Previous work with Yorick [75] demonstrated an automatic vision-based method for centering the elevation and vergence axes which achieved accuracy of this order.

- Vehicle: For the vehicle motion model (Section 3.4), it was necessary to know the scaling between control inputs and movements of the steering axis and driving wheel. The steering axis was well calibrated as standard in the OxNav controller, so that a steering demand of 10° reliably represented an actual 10° movement. The driving wheel scaling was calibrated by hand to two significant figures in a measured run where the odometry count was noted. This is a quantity which could actually be estimated by the robot during its navigation with a change to the map-building and localisation algorithm, since there need only be one quantity determining overall scale in the system, and it is sensible for this to come from the head measurements. This would eliminate a potential source of bias in estimations if calibration is incorrect. As the system stands, the drive scaling is used as well, although in the estimation processes is is given less weight than the head measurements since it is assumed not to be well known.

3

Robot Models and Notation

This chapter describes the mathematical models which have been constructed of the cameras, active head and vehicle which make up the robot used in this research.

It has been considered important to use models which are realistic when reasonably possible. In particular, there is no reason to make approximations with models when the true situation is not difficult to analyse and a large gain is not made by approximating. A case where an approximation can be useful is where it allows some previously developed theory to be applied easily, and we will give an example of this in Section 4.3. However, it is felt that simplifications are sometimes used too quickly in situations where they are not necessary.

3.1 Notation

The notation used in this thesis broadly follows conventions in computer vision, but we will describe the exact meanings of certain representations here, particularly with regard to vectors.

3.1.1 Vectors

When we write a vector like $\mathbf{a}$ in bold type but with no superscript, we mean a physical vector in 3D space independent of any frames of reference we may have introduced. With vectors of this type we can write vector equations such as

$$\mathbf{a} = \mathbf{b} + \mathbf{c}$$

without specifying a coordinate system. This equation simply represents the geometric arrangement in Figure 3.1.

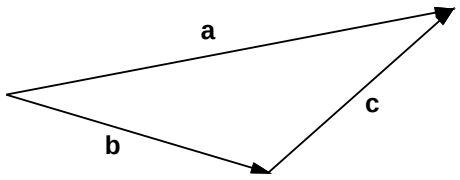


Figure 3.1: The vector sum $\mathbf{a} = \mathbf{b} + \mathbf{c}$.

Subscripts are sometimes used to differentiate between different physical vectors, though ones with a similar role — for instance $\mathbf{c}_L$ and $\mathbf{c}_R$ on the left and right sides of the active head in Section 3.3.

3.1.2 Reference Frames

A frame of reference, denoted in this thesis by a capital letter such as F or a capital letter and a number such as $C0$, is an imaginary alignment of a set of xyz axes in Euclidean 3D space. When vectors are being considered, it is only the *orientation* of this set of axes which is important, not the position of its origin, since vectors are differences between positions, rather than absolute positions in a given reference frame.

Superscripts are used to relate vectors to reference frames. By the notation $\mathbf{a}^F$ we mean the mathematical 3-component vector whose components constitute the spatial vector $\mathbf{a}$ in the reference frame F:

$$\mathbf{a}^F = \begin{pmatrix} a_x^F \\ a_y^F \\ a_z^F \end{pmatrix} .$$

Hence, for example, given the spatial equation $\mathbf{a} = \mathbf{b} + \mathbf{c}$, we can deduce a component equation $\mathbf{a}^F = \mathbf{b}^F + \mathbf{c}^F$ in a particular frame of interest.

It will invariably be possible to represent all physical quantities of interest as vectors. For instance, with a point in space defined as an origin, such as the optic centre of a camera, locations of features in the scene relative to this point can be represented by the vectors from the origin to the features (their *position vectors*). Sometimes, though, in situations like this we will talk about the absolute positions of features in a reference frame centred at the origin — meaning that the location of the reference axes is specified as well as their orientation. This nomenclature can clarify situations where many vectors emanate from the same origin point. We will, however, return to a vector-based representation and reference frames specified only by their orientations when transformations are to be considered, as this method trivialises their application.

3.1.3 Rotations

We will frequently need to transform a vector from one frame to another — that is to say, given the components of a vector in one frame of reference, calculate its components in another. This transformation will take the form of a rotation matrix. We define $\mathtt{R}^{\mathtt{FG}}$ to be the rotation matrix which produces the components of a vector in frame F from its components in frame G:

$$\mathbf{a}^F = \mathtt{R}^{\mathtt{FG}}\mathbf{a}^G .$$

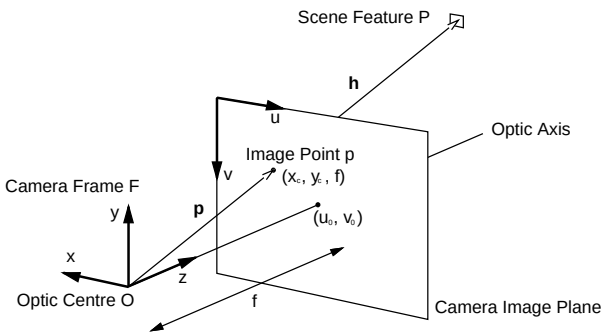


Figure 3.2: The pinhole camera model with focal length f and principal point (u_0, v_0).

Note how the two frame superscripts of $\mathtt{R}$ are seen to be "adjacent" to their respective vectors in the equation.

The reverse transformation is easily formulated, since a rotation, as an orthogonal matrix, has an inverse equal to its transpose:

$$\mathbf{a}^G = \mathtt{R}^{\mathtt{GF}}\mathbf{a}^F \ ,$$

where

$$\mathtt{R}^{\mathtt{GF}} = (\mathtt{R}^{\mathtt{FG}})^\top \ .$$

These rotation transformations should not be thought of as any kind of movement of the vector under consideration in the real world — the spatial vector $\mathbf{a}$ is a fixed entity representing a physical quantity. We are simply dealing with the representation of this vector in imagined frames of reference placed in the world.

3.2 Camera Model

Figure 3.2 shows the customary pinhole model of a CCD video camera which was used in this work. For ease of representation, the camera's image plane is shown in front of the optic centre instead of in its true position behind. A vector $\mathbf{h}$ from the optic centre O to a feature P in the scene intersects the image plane at position p, and this is where the feature will be imaged.

A camera-centred coordinate frame F is defined with origin at O, its z axis aligned with the camera's optic axis, and its x and z axes parallel to the image plane in the horizontal and vertical directions respectively. In this frame, $\mathbf{h}$ has components:

$$\mathbf{h}^F = \begin{pmatrix} h_x^F \\ h_y^F \\ h_z^F \end{pmatrix} \ .$$

The position vector $\mathbf{p}$ of the image point p is parallel to $\mathbf{h}$, but shorter by a factor f/h_z^F. In frame F, the components of $\mathbf{p}$ are:

$$\mathbf{p}^F = \begin{pmatrix} x_c \\ y_c \\ f \end{pmatrix} = \frac{f}{h_z^F}\mathbf{h}^F \ .$$

x_c and y_c are the coordinates of p on the image plane. Now the real image plane is made up of pixels which are spaced at k_u pixels m^{-1} and k_v pixels m^{-1} in the horizontal and vertical directions respectively. (u_0, v_0) is the pixel position of the point where the optic axis crosses the plane. Hence the pixel coordinates of the point (x_c, y_c, f) on the plane are given by

$$u = u_0 - k_u x_c$$

and

$$v = v_0 - k_v y_c.$$

We can write these equations in the form:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} -k_u & 0 & u_0/f \\ 0 & -k_v & v_0/f \\ 0 & 0 & 1/f \end{bmatrix} \begin{pmatrix} x_c \\ y_c \\ f \end{pmatrix},$$

or

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} -k_u & 0 & u_0/f \\ 0 & -k_v & v_0/f \\ 0 & 0 & 1/f \end{bmatrix} \mathbf{p}_F \ .$$

Vector $\mathbf{p}^F$ is parallel to $\mathbf{h}^F$. We can therefore write:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \propto \begin{bmatrix} -fk_u & 0 & u_0 \\ 0 & -fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{h}^F \ ,$$

where we have multiplied through by the constant f to obtain the familiar form of the perspective projection equation:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \propto \mathtt{C}\mathbf{h}^F \ , \tag{3.1}$$

where

$$\mathtt{C} = \begin{bmatrix} -fk_u & 0 & u_0 \\ 0 & -fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

is the *camera calibration matrix*, encoding information about the camera's intrinsic parameters f, k_u, k_v, u_0 and v_0. This equation relates the 3D location of a scene point in the camera reference frame to the homogeneous vector $\begin{pmatrix} u & v & 1 \end{pmatrix}^\top$ describing the pixel location of its image. The homogeneous notation is used because it makes it easy to invert the equation to obtain the inverse relation:

$$\mathbf{h}^F \propto \mathtt{C}^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \ , \tag{3.2}$$

with

$$\mathtt{C}^{-1} = \begin{bmatrix} \frac{-1}{fk_u} & 0 & \frac{u_0}{fk_u} \\ 0 & \frac{-1}{fk_v} & \frac{v_0}{fk_v} \\ 0 & 0 & 1 \end{bmatrix} \ .$$

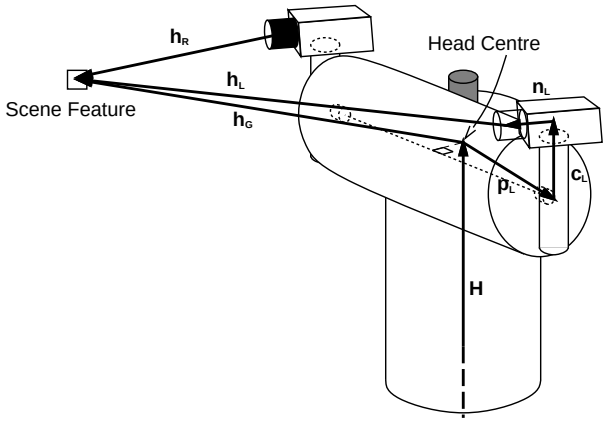


Figure 3.3: A vector model of the Yorick active head. **H** goes vertically from the ground plane to the head centre — the point where the pan axis meets the horizontal plane containing the elevation axis. $\mathbf{p}_L$, $\mathbf{c}_L$ and $\mathbf{n}_L$ combine to represent the offset from the head centre to the left camera's optic centre, with $\mathbf{p}_R$, $\mathbf{c}_R$ and $\mathbf{n}_R$ (not shown) doing the same for the right camera. $\mathbf{h}_L$ and $\mathbf{h}_R$ are vectors from the optic centres to a feature in the scene. $\mathbf{h}_G$ is a calculated vector from the head centre to the feature.

This allows the vector location up to a scale factor of a scene feature to be calculated from its image position (assuming knowledge of `C`) — meaning that the point's 3D location is known to be somewhere along the line defined by all possible lengths of **h**.

3.3 Active Head Model

Figure 3.3 shows the vector model used to represent the active head. The geometry of the head is complicated by the fact that there are small offsets between the axes of rotation, and care has been taken to represent the effect of these. The purpose of the model is to enable the calculation of forward and backward projections through the head system; specifically:

1. The image coordinates at which a known 3D point in the scene will be found in the left and right cameras.
2. The head joint angles necessary to fixate a known 3D point.
3. The 3D position of a point whose image coordinates are known.

3.3.1 The Head Centre

A point on the active head is defined which does not move when the head rotates and which is therefore fixed relative to the vehicle which carries the head. This point is called the "head centre", and is the intersection of the vertical pan axis with the horizontal plane which always contains the elevation axis. The head centre is a reference point between the two cameras which can be used as the origin of measurements from a hypothetical cyclopean sensor. The vector $\mathbf{h}_G$ will represent the displacement from the head centre to a feature of interest in the scene.

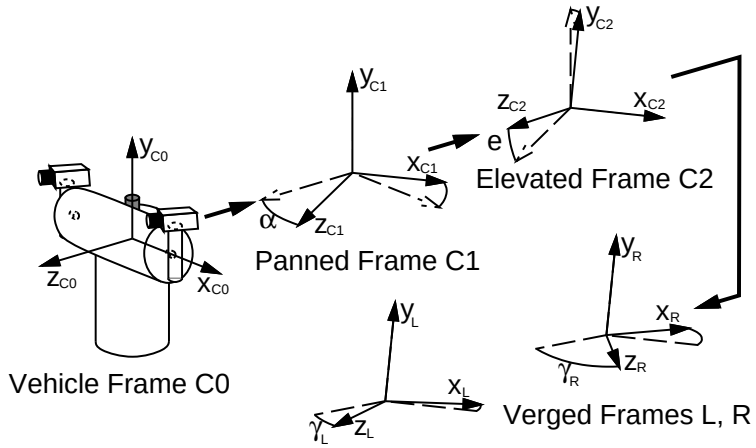


Figure 3.4: The frames of reference used in the head model.

The head's encoders provide continuous and accurate information on the angles of its rotation axes. Knowledge of these angles means that is is always possible to calculate the location and orientation of the two cameras relative to the head centre.

3.3.2 Reference Frames

Referring back to Figure 2.5 showing the Yorick's rotation axes, and to Figure 3.4, we introduce the reference frames used when describing the head. When the head angles are all zero, both cameras will point directly forwards and all of these frames will be aligned with $C0$. The reason for introducing these frames is that each of the vectors in the head model in Figure 3.3 can be represented in a simple way in one of them, being either constant in the case of the internal head vectors $\mathbf{H}$, $\mathbf{p}_L$, $\mathbf{p}_R$, $\mathbf{c}_L$, $\mathbf{c}_R$, $\mathbf{n}_L$ and $\mathbf{n}_R$, or in the representation in which they will be useful in the case of the "end result" vectors $\mathbf{h}_L$, $\mathbf{h}_R$ and $\mathbf{h}_G$.

- $C0$ is the *vehicle frame* which is fixed relative to the vehicle. Its z axis lies in the forward direction, with the y-axis pointing up and the x axis to the left. In this frame:

$$\mathbf{H}^{C0} = \begin{pmatrix} 0 \\ H \\ 0 \end{pmatrix} \quad , \quad \mathbf{h}_G^{C0} = \begin{pmatrix} h_{Gx}^{C0} \\ h_{Gy}^{C0} \\ h_{Gz}^{C0} \end{pmatrix} .$$

- $C1$ is the *panned frame* which is fixed relative to the parts of the head that move when the pan angle changes. Its y axis stays vertical, and its x and z axes remain in the horizontal plane, with its x axis always parallel to the head's elevation axis. It is horizontally rotated with respect to $C0$ by the pan angle α. In this frame:

$$\mathbf{p}_L^{C1} = \begin{pmatrix} I/2 \\ 0 \\ p \end{pmatrix} \quad , \quad \mathbf{p}_R^{C1} = \begin{pmatrix} -I/2 \\ 0 \\ p \end{pmatrix} .$$

- $C2$ is the *elevated frame* which is fixed relative to the parts of the head that move when the elevation changes. Its y axis stays parallel to the two vergence axes, and its x axis is aligned with the elevation axis. It is vertically rotated with respect to $C1$ by elevation angle e. In this frame:

$$\mathbf{c}_L^{C2} = \begin{pmatrix} 0 \\ c \\ 0 \end{pmatrix} \quad , \quad \mathbf{c}_R^{C2} = \begin{pmatrix} 0 \\ c \\ 0 \end{pmatrix} \ .$$

- L and R are the left and right camera frames respectively, fixed relative to the two cameras. Their z axes are aligned with the camera optic axes, and their x and y axes lie parallel to their image planes. These frames are rotated with respect to $C2$ by vergence angles γ_L and γ_R respectively. In these frames:

$$\mathbf{n}_L^L = \begin{pmatrix} 0 \\ 0 \\ n \end{pmatrix} \quad , \quad \mathbf{h}_L^L = \begin{pmatrix} h_{Lx}^L \\ h_{Ly}^L \\ h_{Lz}^L \end{pmatrix} \ ,$$

$$\mathbf{n}_R^R = \begin{pmatrix} 0 \\ 0 \\ n \end{pmatrix} \quad , \quad \mathbf{h}_R^R = \begin{pmatrix} h_{Rx}^R \\ h_{Ry}^R \\ h_{Rz}^R \end{pmatrix} \ .$$

H, I, p, c, and n are scalar distances representing the dimensions of the head:

- H is the vertical distance of the head centre above the ground plane.
- I the is horizontal distance between the vergence axes (the *inter-ocular* distance between the camera optic centres if both cameras are facing forwards).
- p is the horizontal offset between the pan and elevation axes.
- c is the offset along either vergence axis between the intersections with the elevation axis and the camera optic axis.
- n is the offset along either optic axis between the camera optic centre and the intersection with the vergence axis.

The rotation matrices transforming between these reference frames have trivial forms due to their simple relationships:

$$\mathtt{R}^{\mathtt{C10}} = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix}$$

$$\mathtt{R}^{\mathtt{C21}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos e & -\sin e \\ 0 & \sin e & \cos e \end{bmatrix}$$

$$\mathtt{R}^{\mathtt{LC2}} = \begin{bmatrix} \cos\gamma_L & 0 & -\sin\gamma_L \\ 0 & 1 & 0 \\ \sin\gamma_L & 0 & \cos\gamma_L \end{bmatrix}$$

$$\mathtt{R}^{\mathtt{RC2}} = \begin{bmatrix} \cos\gamma_R & 0 & -\sin\gamma_R \\ 0 & 1 & 0 \\ \sin\gamma_R & 0 & \cos\gamma_R \end{bmatrix}$$

3.3.3 Using the Model

With this framework in place, the expressions required from the model can easily be calculated. Referring to Figure 3.3, we may form the two spatial vector equations:

$$\mathbf{h}_G = \mathbf{p}_L + \mathbf{c}_L + \mathbf{n}_L + \mathbf{h}_L \ , \tag{3.3}$$

$$\mathbf{h}_G = \mathbf{p}_R + \mathbf{c}_R + \mathbf{n}_R + \mathbf{h}_R \ . \tag{3.4}$$

Finding the Image Coordinates of a Known 3D Point

Consider first the case where $\mathbf{h}_G^{C0}$, the location of a 3D point relative the the head centre in the vehicle coordinate frame, is known, and we wish to calculate its image position in the left and right cameras (assuming that it is in their fields of view): for the left camera, we form Equation 3.3 in the left camera coordinate frame L:

$$\mathbf{h}_L^L = \mathbf{h}_G^L - \mathbf{p}_L^L - \mathbf{c}_L^L - \mathbf{n}_L^L \ .$$

Rephrasing this equation with transformations so that vectors can be used in their natural reference frames:

$$\mathbf{h}_L^L = \mathtt{R}^{\mathtt{LC0}}\mathbf{h}_G^{C0} - \mathtt{R}^{\mathtt{LC1}}\mathbf{p}_L^{C1} - \mathtt{R}^{\mathtt{LC2}}\mathbf{c}_L^{C2} - \mathbf{n}_L^L \ ,$$

or, dividing the transformations into their known, simple, components:

$$\mathbf{h}_L^L = \mathtt{R}^{\mathtt{LC2}}(\mathtt{R}^{\mathtt{C21}}(\mathtt{R}^{\mathtt{C10}}\mathbf{h}_G^{C0} - \mathbf{p}_L^{C1}) - \mathbf{c}_L^{C2}) - \mathbf{n}_L^L \ . \tag{3.5}$$

For the right camera:

$$\mathbf{h}_R^R = \mathtt{R}^{\mathtt{RC2}}(\mathtt{R}^{\mathtt{C21}}(\mathtt{R}^{\mathtt{C10}}\mathbf{h}_G^{C0} - \mathbf{p}_R^{C1}) - \mathbf{c}_R^{C2}) - \mathbf{n}_R^R \ . \tag{3.6}$$

These are the equations in the form in which they will be used. They allow $\mathbf{h}_L^L$ and $\mathbf{h}_R^R$ to be found — these are the vectors from the left and right optic centres to the scene point in the left and right camera coordinate frames respectively. From Equation 3.1 the point's image coordinates can be determined:

$$\begin{pmatrix} u_L \\ v_L \\ 1 \end{pmatrix} \propto \mathtt{C}\mathbf{h}_L^L \quad , \quad \begin{pmatrix} u_R \\ v_R \\ 1 \end{pmatrix} \propto \mathtt{C}\mathbf{h}_R^R \ . \tag{3.7}$$

Finding the Head Angles to Fixate a Known Point

We will frequently need to fixate a 3D point at a known location $\mathbf{h}_G^{C0}$ relative the the head centre — that is, drive the active head so that both cameras point directly towards it. To calculate the head angles required, we must first decide how to deal with the redundancy introduced by the head geometry: there are many sets of angles possible. We choose always to use symmetric fixation: the vergence angles are required to be equal and opposite.

Symmetric fixation is permitted by turning the pan axis to face the scene point, so that in the $C1$ panned frame $\mathbf{h}_G$ lies in the vertical yz plane: thus $\mathbf{h}_{Gx}^{C1} = 0$. Writing

$$\mathbf{h}_G^{C1} = \mathtt{R}^{\mathtt{C10}}\mathbf{h}_G^{C0}$$

and solving for α in the top line of the vector equation, we obtain:

$$\alpha = \tan^{-1}\frac{h_{Gx}^{C0}}{h_{Gz}^{C0}} \tag{3.8}$$

(where α is forced into the range $-\pi \rightarrow \pi$).

To find e, γ_L and γ_R, we observe that at fixation, both cameras' optic axes should pass through the scene point; therefore, $\mathbf{h}_L^L$ and $\mathbf{h}_R^R$ are both non-zero only in the z coordinate: $h_{Lx}^L = h_{Ly}^L = h_{Rx}^R = h_{Ry}^R = 0$.

Forming Equation 3.3 in frame L, we obtain

$$\mathbf{h}_L^L = \mathtt{R}^{\mathtt{LC2}}(\mathtt{R}^{\mathtt{C21}}(\mathtt{R}^{\mathtt{C10}}\mathbf{h}_G^{C0} - \mathbf{p}_L^{C1}) - \mathbf{c}_L^{C2}) - \mathbf{n}_L^L \ .$$

as in Equation 3.5. Setting $h_{Ly}^L = 0$ and solving for e gives the result:

$$e = \tan^{-1}\frac{h_{Gy}^{C1}}{h_{Gz}^{C1} - p} - \sin^{-1}\frac{c}{\sqrt{{h_{Gy}^{C1}}^2 + (h_{Gz}^{C1} - p)^2}} \ . \tag{3.9}$$

(The components of $\mathbf{h}_G^{C1}$ are known since we already know α.) Finally, to find the vergence angles, e is used to form $\mathbf{h}_L^{C2}$; then, in the equation

$$\mathbf{h}_L^L = \mathtt{R}^{\mathtt{LC2}}\mathbf{h}_L^{C2} \ ,$$

h_{Lx}^L is set to zero to find γ_L as:

$$\gamma_L = \tan^{-1}\frac{h_{Lx}^{C2}}{h_{Lz}^{C2}} \ . \tag{3.10}$$

Trivially,

$$\gamma_R = -\gamma_L \ . \tag{3.11}$$

Finding a 3D Point from Known Image Coordinates

The third case to be tackled is where the left and right image coordinates u_L, v_L, u_R, and v_R of a point are known and we wish to calculate the point's 3D position.

This time, for the left camera, Equation 3.3 is formed in the vehicle coordinate frame $C0$:

$$\mathbf{h}_G^{C0} = \mathbf{p}_L^{C0} + \mathbf{c}_L^{C0} + \mathbf{n}_L^{C0} + \mathbf{h}_L^{C0} \ .$$

Rephrasing with transformations to get vectors in their natural frames:

$$\mathbf{h}_G^{C0} = \mathtt{R}^{\mathtt{C01}}\mathbf{p}_L^{C1} + \mathtt{R}^{\mathtt{C02}}\mathbf{c}_L^{C2} + \mathtt{R}^{\mathtt{C0L}}\mathbf{n}_L^L + \mathtt{R}^{\mathtt{C0L}}\mathbf{h}_L^L \ .$$

or

$$\mathbf{h}_G^{C0} = \mathtt{R}^{\mathtt{C01}}(\mathbf{p}_L^{C1} + \mathtt{R}^{\mathtt{C12}}(\mathbf{c}_L^{C2} + \mathtt{R}^{\mathtt{C2L}}\mathbf{n}_L^L)) + \mathtt{R}^{\mathtt{C01}}\mathtt{R}^{\mathtt{C12}}\mathtt{R}^{\mathtt{C2L}}\mathbf{h}_L^L \ . \tag{3.12}$$

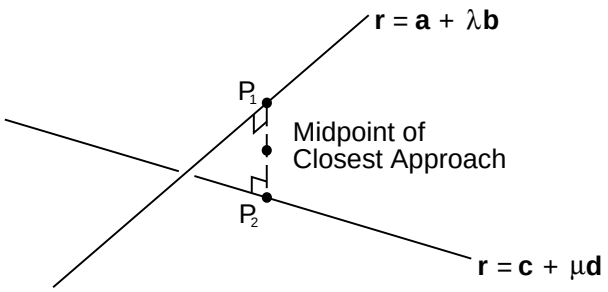


Figure 3.5: The closest intersection of skew lines calculated as the midpoint of their closest approach.

For the right camera:

$$\mathbf{h}_G^{C0} = \mathtt{R}^{\mathtt{C01}}(\mathbf{p}_R^{C1} + \mathtt{R}^{\mathtt{C12}}(\mathbf{c}_R^{C2} + \mathtt{R}^{\mathtt{C2R}}\mathbf{n}_R^R)) + \mathtt{R}^{\mathtt{C01}}\mathtt{R}^{\mathtt{C12}}\mathtt{R}^{\mathtt{C2R}}\mathbf{h}_R^R \ . \tag{3.13}$$

Now we have two expressions for $\mathbf{h}_G^{C0}$, but neither is sufficient on its own. The reason is that $\mathbf{h}_L^L$ and $\mathbf{h}_R^R$, the vectors from the camera optic centres to the scene point, are not known exactly but only up to a scale factor from the inverted projection equations:

$$\mathbf{h}_L^L \propto \mathtt{C}^{-1}\begin{pmatrix} u_L \\ v_L \\ 1 \end{pmatrix} \quad , \quad \mathbf{h}_R^R \propto \mathtt{C}^{-1}\begin{pmatrix} u_R \\ v_R \\ 1 \end{pmatrix} \ . \tag{3.14}$$

Equations 3.12 and 3.13 are therefore of the form:

$$\begin{aligned} \mathbf{h}_G^{C0} &= \mathbf{a}^{CO} + \lambda\mathbf{b}^{C0} \ , \\ \mathbf{h}_G^{C0} &= \mathbf{c}^{C0} + \mu\mathbf{d}^{C0} \ , \end{aligned}$$

where

$$\begin{aligned} \mathbf{a}^{CO} &= \mathtt{R}^{\mathtt{C01}}(\mathbf{p}_L^{C1} + \mathtt{R}^{\mathtt{C12}}(\mathbf{c}_L^{C2} + \mathtt{R}^{\mathtt{C2L}}\mathbf{n}_L^L)) \ , \\ \mathbf{b}^{C0} &= \mathtt{R}^{\mathtt{C01}}\mathtt{R}^{\mathtt{C12}}\mathtt{R}^{\mathtt{C2L}}\mathtt{C}^{-1}\begin{pmatrix} u_L \\ v_L \\ 1 \end{pmatrix} \ , \\ \mathbf{c}^{C0} &= \mathtt{R}^{\mathtt{C01}}(\mathbf{p}_R^{C1} + \mathtt{R}^{\mathtt{C12}}(\mathbf{c}_R^{C2} + \mathtt{R}^{\mathtt{C2R}}\mathbf{n}_R^R)) \ , \\ \mathbf{d}^{C0} &= \mathtt{R}^{\mathtt{C01}}\mathtt{R}^{\mathtt{C12}}\mathtt{R}^{\mathtt{C2R}}\mathtt{C}^{-1}\begin{pmatrix} u_R \\ v_R \\ 1 \end{pmatrix} \ . \end{aligned}$$

λ and μ are undetermined scalars which are greater than zero. The situation is that each camera measurement provides a line which the scene point must lie on. Clearly, the actual position of the scene point will be found as the intersection of the two lines. In the presence of pixel quantisation and measurement errors, however, the calculated lines will be skew and will not actually intersect exactly. Their "best intersection" is therefore calculated as the midpoint of their closest approach, as depicted in Figure 3.5. The coordinates of this point are easily calculated in terms of the components of vectors $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$ by evaluating λ and μ for the points P_1 and P_2 (the criterion being that the vector from P_1 to P_2 must be perpendicular to both lines) and substituting into either of the line equations above.

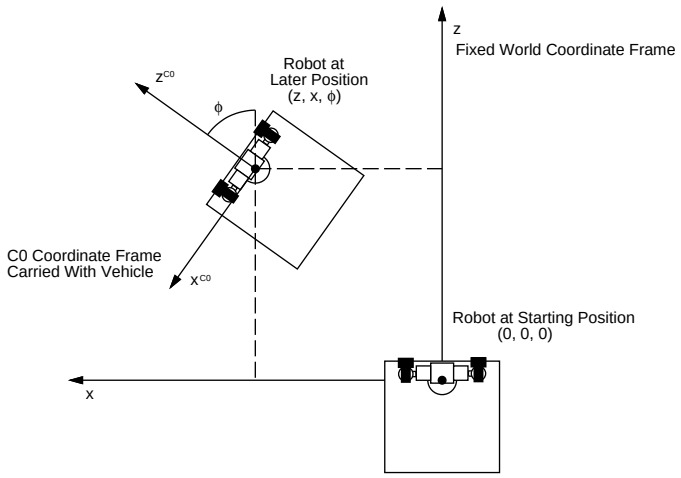


Figure 3.6: The vehicle's location in the world coordinate frame is specified with the coordinates (z, x, ϕ). The $C0$ coordinate frame is carried with the vehicle.

3.4 Vehicle Model

We have assumed that the robot vehicle moves on a perfect ground-plane at all times. A world coordinate frame W is defined such that its x and z axes lie in the ground plane, and its y axis points vertically upwards. The robot's position on the ground is specified by the coordinates (z, x, ϕ), where z and x are the world-frame coordinates of the head centre (its position in the y direction being constant), and ϕ is the robot's orientation relative to the world z axis.

As shown in Figure 3.6, at the start of its motion the vehicle's location in the world frame is defined to be $z = 0, x = 0, \phi = 0$, and the world and vehicle frames are coincident (we will discuss this further in Chapter 5).

3.4.1 Moving the Robot

As described in the previous chapter, our robot vehicle has three wheels: two of these are free-running but fixed to be forward-facing, and lie along the same axis at the front of the vehicle. The head centre lies vertically above this axis. The third wheel at the rear is steerable and also provides the drive.

With no transverse slipping of the wheels across the floor, setting a steering angle of s at the rear wheel means that points on the vehicle will travel in circular paths about the centre of rotation at the intersection of the wheel axes. From Figure 3.7 we see that the radius of the path that the "head centre" moves in is

$$R = L/\tan(s) \ .$$

The radius of the path that the rear driving wheel moves in is

$$R_d = L/\sin(s) \ .$$

The control inputs to the vehicle which determine its motion are the steering angle s and the velocity v at which the rear wheel is driven. Consider a period of time Δt in which

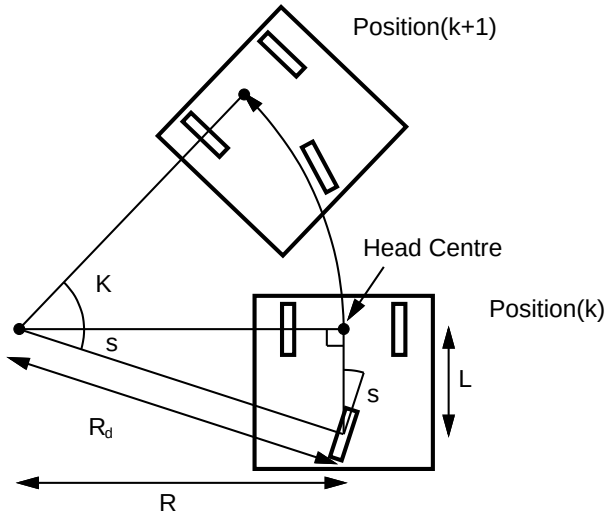


Figure 3.7: With the angle of the vehicle's single rear steering wheel set to $s(k)$, the "head centre", lying centrally above the fixed axis of the front wheels, moves in a path of radius $R(k) = L/\tan(s(k))$. L is the vehicle's constant wheelbase.

both v and s are held at constant values: since the radius of the circular path along which the driving wheel moves is R_d, the angle in radians through which the vehicle moves along its circular trajectory is

$$K = v\Delta t/R_d \ .$$

We can calculate the change in location of the head centre:

$$z(t+\Delta t) = z(t) + R\left(\cos\phi(t)\sin K + \sin\phi(t)(\cos K - 1)\right) \quad (3.15)$$

$$x(t+\Delta t) = x(t) + R\left(\sin\phi(t)\sin K + \cos\phi(t)(1-\cos K)\right) \quad (3.16)$$

$$\phi(t+\Delta t) = \phi(t) + K \quad (3.17)$$

These are exact expressions and do not require Δt to be small. They are equally valid for backward vehicle motions if v has a negative value. Note that for the special case where $s = 0$ (straight line motion) we must use a limiting form of these equations since R and R_d become infinite:

$$z(t+\Delta t) = z(t) + v\Delta t\cos\phi(t) \quad (3.18)$$

$$x(t+\Delta t) = x(t) + v\Delta t\sin\phi(t) \quad (3.19)$$

$$\phi(t+\Delta t) = \phi(t) \quad (3.20)$$

3.4.2 Errors in Motion Estimates

Clearly, if we were able to rely precisely on these equations to calculate how the robot was moving after being sent control inputs v and s, it would not be necessary to use vision or other sensors to provide localisation information. The robot will, however, not move exactly in the way predicted by the equations due to a number of possible factors: these

include slipping of the wheels and incorrect calibration of the vehicle controller. The result will be that although the expressions above will represent our best estimate of the vehicle's position, we must assign an uncertainty to this estimate.

The uncertainty is modelled as Gaussian variation in the effective control inputs v and s from the demanded values. In case of the steering angle s, the variation was given a constant standard deviation σ_s of around 8°, but with the velocity input v it was more appropriate to assign a standard deviation which was proportional to the velocity demand itself — meaning that large motions are more uncertain than small ones. We used $\sigma_v = \sigma_f v$, with $\sigma_f \approx 0.15$.

Naming the estimated position vector $\mathbf{f}_v$ and the control vector $\mathbf{u}$:

$$\mathbf{f}_v = \begin{pmatrix} z(t+\Delta t) \\ x(t+\Delta t) \\ \phi(t+\Delta t) \end{pmatrix} \quad , \quad \mathbf{u} = \begin{pmatrix} v \\ s \end{pmatrix} \ , \tag{3.21}$$

we can calculate $\mathtt{Q}$, the covariance matrix for $\mathbf{f}_v$, using the standard formula for transfer of uncertainty at first order:

$$\mathtt{Q} = \frac{\partial \mathbf{f}_v}{\partial \mathbf{u}} \mathtt{U} \frac{\partial \mathbf{f}_v}{\partial \mathbf{u}}^\top \ , \tag{3.22}$$

where $\frac{\partial \mathbf{f}_v}{\partial \mathbf{u}}$ is the Jacobian between $\mathbf{f}_v$ and $\mathbf{u}$, and $\mathtt{U}$ is the covariance matrix of $\mathbf{u}$:

$$\frac{\partial \mathbf{f}_v}{\partial \mathbf{u}} = \begin{bmatrix} \frac{\partial z(t+\Delta t)}{\partial v} & \frac{\partial z(t+\Delta t)}{\partial s} \\ \frac{\partial x(t+\Delta t)}{\partial v} & \frac{\partial x(t+\Delta t)}{\partial s} \\ \frac{\partial \phi(t+\Delta t)}{\partial v} & \frac{\partial \phi(t+\Delta t)}{\partial s} \end{bmatrix} \quad , \quad \mathtt{U} = \begin{bmatrix} {\sigma_v}^2 & 0 \\ 0 & {\sigma_s}^2 \end{bmatrix} \ . \tag{3.23}$$

Explicitly,

$$\begin{aligned}
\frac{\partial z(t+\Delta t)}{\partial v} &= R\frac{\partial K}{\partial v}(\cos\phi\cos K - \sin\phi\sin K) \\
\frac{\partial z(t+\Delta t)}{\partial s} &= R\frac{\partial K}{\partial s}(\cos\phi\cos K - \sin\phi\sin K) + \frac{\partial R}{\partial s}(\cos\phi\sin K + \sin\phi(\cos K - 1)) \\
\frac{\partial x(t+\Delta t)}{\partial v} &= R\frac{\partial K}{\partial v}(\sin\phi\cos K - \cos\phi\sin K) \\
\frac{\partial x(t+\Delta t)}{\partial s} &= R\frac{\partial K}{\partial s}(\sin\phi\cos K - \cos\phi\sin K) + \frac{\partial R}{\partial s}(\sin\phi\sin K + \cos\phi(1 - \cos K)) \\
\frac{\partial \phi(t+\Delta t)}{\partial v} &= \frac{\partial K}{\partial v} \\
\frac{\partial \phi(t+\Delta t)}{\partial s} &= \frac{\partial K}{\partial s} \ ,
\end{aligned} \tag{3.24}$$

where:

$$\frac{\partial K}{\partial v} = \frac{\Delta t}{R} \quad , \quad \frac{\partial K}{\partial s} = \frac{v\Delta t\cos\phi}{L} \quad , \quad \frac{\partial R}{\partial s} = \frac{-L}{\sin s^2} \ . \tag{3.25}$$

There is again a special limiting case when $s = 0$:

$$\frac{\partial \mathbf{f}_v}{\partial \mathbf{u}} = \begin{bmatrix} \Delta t\cos\phi & \frac{-v^2\Delta t^2\sin\phi}{2L} \\ \Delta t\sin\phi & \frac{-v^2\Delta t^2\cos\phi}{2L} \\ 0 & \frac{v\Delta t}{L} \end{bmatrix} \ . \tag{3.26}$$

The measure of uncertainty in vehicle position provided by this expression will turn out to be very important when we look at incorporating odometry information with head measurements in Chapter 5.

As mentioned at the end of Section 2.8, this simple noise model could be extended to account for the more systematic error source of incorrect calibration of the robot odometry: quantities such as the velocity demand/output scaling and true steering "zero" position could be estimated continuously by augmenting the system state vector of Chapter 5 to explicitly include them. From initial estimates, their values could be improved over a robot journey from the information provided by feature measurements. The experiments of the later chapters provide some evidence that this might be necessary. However, a line must be drawn somewhere in the complexity of the system (there are many higher-order effects which could be estimated as well) and this noise model generally performs well.

4

Vision Tools, and an Example of their Use

This chapter has two clear parts. In Sections 4.1 and 4.2, we introduce and discuss the vision capabilities and tools used in all the work in the rest of the thesis, concentrating on the use of arbitrary scene features as landmarks and the concept of fixation.

Then, in Section 4.3, a preliminary use of these methods for navigation is presented, in an active implementation of a the 8-point structure from motion algorithm. This approach makes use of the interesting frontal plane and gazesphere ideas, but as will be discussed has disadvantages for long-term navigation which will be tackled in Chapter 5.

4.1 Scene Features

Our map-building and localisation system is based on discrete features, like the majority of robot navigation systems. Discrete features are easier to represent than continuous objects or regions, and provide unambiguous information for localisation. The spatial density of features used can differ depending on how well it is required to represent the world, and individual ones can easily be added or deleted as necessary. (Chapter 7 will discuss the failings of feature-based methods).

The features which will make up the sparse map used for localisation in this work need to be "landmarks" for the robot: they must be reliably and repeatedly detectable and measurable, and preferably from a wide range of robot positions. This differs slightly from the requirements of the features used in structure from motion algorithms (see the discussion in Section 1.2.1) where it is not usually intended that the same features (usually "corners" or line segments) be detected and matched over long periods — indeed this is typically not possible because with a passive camera features can go out of view very quickly. What

properties do landmarks need to have?

- They must be features present in a normal environment so that no artificial interference such as positioning of beacons is required.
- They must be stationary in the scene (or potentially have known motions, in work beyond the scope of this thesis).
- They must be easily identifiable from a variety of ranges and angles, and not frequently be occluded.

It was decided to restrict landmarks to being stationary, point features, and to use simple image patches to represent them, with matching to be performed using normalised sum-of-squared-difference correlation. The patches used are larger than those usually representing corner features in structure from motion systems: 15×15 pixels rather than 3×3 or 5×5. The reason for this choice is that with larger patches the chances of mismatches are reduced due to their greater distinguishability. Having larger patches makes processing slower than otherwise, but the relatively small number of features used and the fact that with the active approach only one feature is observed at a time means that this is not a problem. The key to our approach is to have a small number of very reliable landmarks rather than a large number of transient ones.

4.1.1 Detecting Features

To detect good features automatically in an image, the operator suggested by Shi and Tomasi [85] has been used. This feature detector is very similar to the Harris corner detector [37], but Shi and Tomasi apply it to patches of a large size similar to that in this work. They derive the operator showing that it finds features which will be intrinsically easy to track using their gradient-based search method. Its function is to find image patches which will make good features due to their high variation in intensity — these stand out well from their surroundings.

To evaluate the "interest" value of a particular patch of an image, first the horizontal and vertical gradients of image intensity, g_x and g_y respectively, are calculated at each pixel position. The 2×2 matrix $\mathtt{Z}$ is then formed, where:

$$\mathtt{Z} = \sum_{\text{patch}} \begin{bmatrix} {g_x}^2 & g_x g_y \\ g_y g_x & {g_y}^2 \end{bmatrix} .$$

The two eigenvalues λ_1 and λ_2 of $\mathtt{Z}$ are found. The patch has a high interest value if the *smaller* of λ_1 and λ_2 is large. Cases where just one of the eigenvalues is large mean that the patch has a large interest score in one image direction, but not in the perpendicular direction — the patch contains an edge-like feature, which is not useful as a point landmark. However, this does mean that the operator can be used as a combined corner and edge detector as in [37].

To find patches which will make good new features, therefore, the operator is scanned over an image, and the patches giving the largest small eigenvalues of $\mathtt{Z}$ are chosen. The algorithm is inexpensive: only first-order image gradients are required, and an efficient scanning implementation can be achieved — to form the sums $\sum {g_x}^2$, $\sum g_x g_y$, $\sum g_y g_x$ and $\sum {g_y}^2$

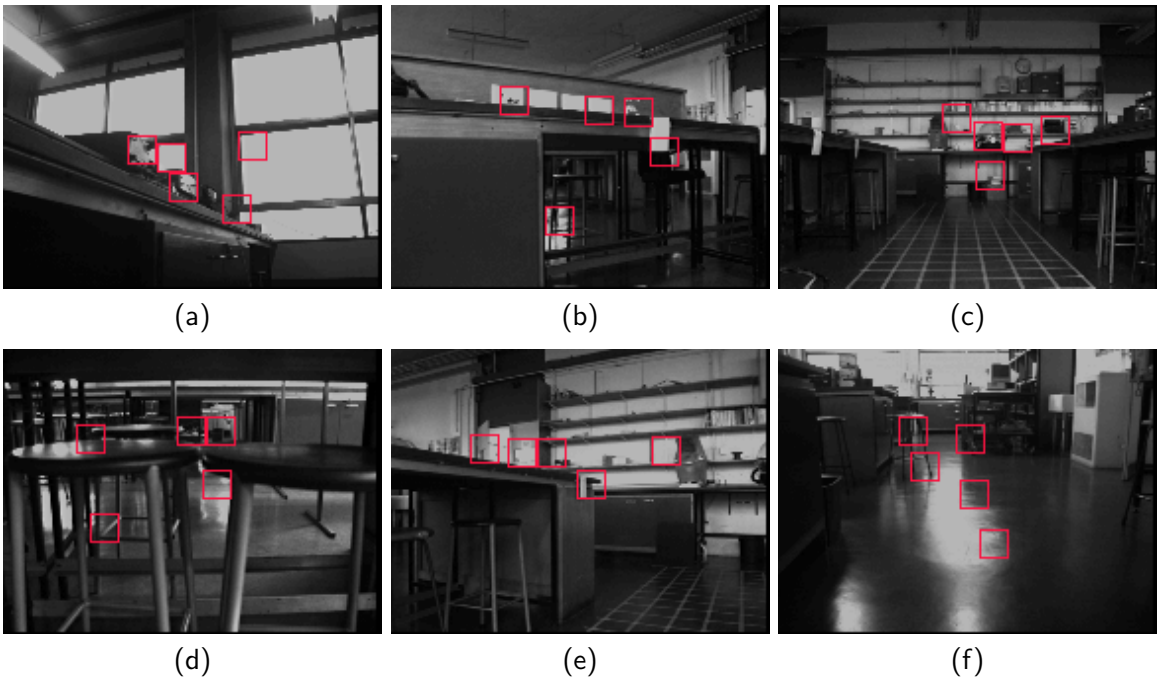


Figure 4.1: The best five feature patches automatically detected in the central region of different views of the laboratory. The views show potential problems with occlusion boundaries (b, d, e) and reflections (d, f) which lead to the selection of patches not corresponding to stationary point features.

at a new patch position, incremental changes can be made to the results from neighbouring locations.

Figure 4.1 shows the best patches found by the operator in different views of the laboratory. The central region only of each image has been searched, and the five best non-overlapping patches are shown. However, it is easy from manual inspection of the images to say that some of the patches will make good landmarks while some will not: they clearly correspond to parts of the image which contain depth discontinuities, meaning that a point landmarks position is not defined, or in other cases to reflections which are not stationary references because they will move as the robot does.

Some previous work, such as that by Shilat *et al.* [86] has attempted to identify "bad" features like these at the detection stage. Their approach is to reject patches which contain very sharp "step" edges, since these are most likely to occur with depth discontinuities or reflective highlights. In our work, and that of others like [85], however, no attempt is made to discern good or bad features at the detection stage: the strategy used is to accept or reject features depending on how well they can be tracked once the robot has started moving. Patches which do not actually correspond to stationary, point features will quickly look very different from a new viewpoint, or will not appear in the position expected from the vehicle motion model, and thus matching will fail. These features can then be deleted from the map.

It should be noted that the patch detection algorithm is run only to find new features,

(a) (b) (c)

Figure 4.2: Elliptical search regions generated for features. The size of the ellipse depends on the uncertainty of the relative positions of the robot and feature, as will be explained in Chapter 5.

which are chosen as the maxima returned. When it is necessary to find a particular feature in a subsequent image, a search is carried out using normalised sum-of-squared-difference: the patch detection algorithm is not used on the new image. This differs from many structure from motion systems where the feature detector is run on each new image: features are found as local maxima of the particular feature-finding operator, and then correlation matching is carried out between the features found in the current and previous images, all of which are local maxima. Some authors [21] have found that requiring all patches matched to be maxima like this is restrictive, and features can be missed which still give good correlation matches. This is a good point when applied to our active system, since it is important to keep track of the same features for as long as possible. Since a search in a particular image is only for one feature at a time, it makes sense to move straight to correlation matching.

One advantage of applying the feature detector to all images is that most detectors return feature locations to sub-pixel accuracy in a natural way. It is more difficult to see how to obtain sub-pixel accuracy from the correlation searching used in our work: interpolation can be used, but the form of the function to fit to image values depends on the image itself. For speed, and since good accuracy has still been obtained, sub-pixel measurements have not been used in our work.

4.1.2 Searching For and Matching Features

It will be shown in Chapter 5 that in the localisation system, whenever a measurement is required of a particular feature, a region can be generated in the left and right images within which the feature should lie with a high probability. This means that only that area of the images need be searched. The region is typically an ellipse centred in the image: some examples are shown in Figure 4.2.

At each position in the region (scanned in raster order), that part of the image is compared with the patch saved to represent the feature being searched for using the following normalised sum-of-squared-difference measure:

$$C = \frac{\sum_{\text{patch}} \left[\frac{g_1 - \bar{g_1}}{\sigma_1} - \frac{g_0 - \bar{g_0}}{\sigma_0} \right]^2}{n} , \tag{4.1}$$

Figure 4.3: Features matched with normalised sum-of-squared-difference correlation showing the wide changes in viewing distance and direction which can be accommodated.

where g_0 and g_1 are the image intensity values at corresponding positions in the saved patch and image patch respectively, and $\bar{g_0}$, $\bar{g_1}$, σ_0 and σ_1 are means and standard deviations of the intensity across the patches. n is the number of pixels in a patch. This expression has the value 0 for a perfect match, with higher values as the difference increases. It was considered essential to use a measure which was normalised with respect to overall intensity changes, since this gives improved matching over the long periods over which the system is required to work, when lighting conditions might change, and also since when working in stereo the two cameras will vary somewhat in their output. What the expression actually measures is the average difference in standard deviation above the patch mean intensity between corresponding pixels in the patches being compared. The best (lowest) value of C found in the search region is accepted as a match if it lies below a threshold (set manually at 0.9).

Figure 4.3 shows matches obtained of some features, giving an impression for the surprising range of viewpoints which can be matched successfully using the simple patch representation of features. However, clearly matching can only be expected to succeed for reasonable robot motions, since the patch representation is intrinsically viewpoint-variant — features look different when viewed from new distances or angles. Therefore, we have defined a criterion for expected visibility based on the differences between the viewpoint from which the feature was initially seen and a new viewpoint. Figure 4.4 shows the situation: $\mathbf{h}_{\text{orig}}$ is the vector from the head centre to the feature when it was initialised, and $\mathbf{h}$ is that from the head centre at a new vehicle position. The feature is expected to be visible if the length ratio $\frac{|\mathbf{h}|}{|\mathbf{h}_{\text{orig}}|}$ is close enough to 1 (in practice between $\frac{5}{7}$ and $\frac{7}{5}$) and the angle difference $\beta = \cos^{-1}((\mathbf{h} \cdot \mathbf{h}_{\text{orig}})/(|\mathbf{h}||\mathbf{h}_{\text{orig}}|))$ is close enough to 0 (in practice less than 45° in magnitude). In the localisation algorithm of Chapter 5, we are in a position to estimate both of these vectors before a measurement is made, and so attempts are made only to

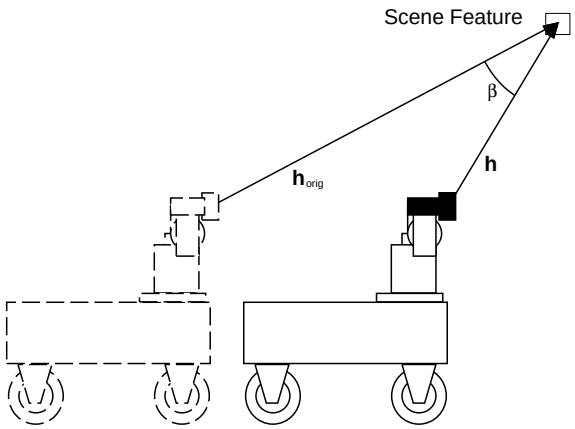


Figure 4.4: The expected visibility of a feature is calculated based on the difference in distance and angle between the viewpoint from which it was initially seen and that from which the current measurement is to be made.

measure features which are expected to be visible. The result is a region of space defined for each feature from which it should be able to be seen. A features which fails to match regularly within this region should be deleted from the map.

It should be noted that the patch representing a certain feature is not updated when a successful match is made and the feature is re-found — one strategy would be to replace the saved patch with the region matched to it in the latest image. This would certainly improve the range of robot movement over which the feature could be tracked, since matching would always be between images taken from closely-spaced viewpoints. However, the problem with this approach is that the patch tends to drift away from its original position over the feature: a small error in locating the feature at each step (due even to just the pixelation limitation) can lead to a large patch movement over many frames. It is essential that landmarks are *stationary* features.

4.1.3 Other Feature Types

Two further feature types have been considered for use in the system:

- Planar patch features: several authors [101, 80, 85] have started to use a representation of point features as small planar regions in the 3D world rather than 2D image patches. If this assumption is made, when an attempted measurement is made of a feature, the estimated transformation between the patch plane and the image plane can be used to warp the saved patch representation into the shape and size in which it will appear from the new viewpoint. Clearly this method has the capability to greatly extend the range of robot motion through which particular features can be seen and measured, and this will benefit localisation systems, where it is always an advantage to see the same landmarks for a long time.

  To initialise a feature of this type, it is necessary to assume that it is part of a planar object (as is certainly approximately the case for many features, such as the markings on a wall), and then to estimate the orientation of that plane. This estimation can

be achieved from stereo views of the patch: the difference in its appearance between the left and right camera views provides the information. However, initial trials have shown that it is difficult to get accurate estimates since the baseline of the active head is relatively small and the patches do not look very different in the left and right views. This problem gets worse the larger the distance from the robot to the feature, but of course with distant features it is less important to know the plane orientation accurately since views of these features will change relatively little with robot motion.

Shi and Tomasi [85] track features as simple patches, but use affine warpings of the patches to perform checks on whether features are still "good" in the sense of being stationary without depth discontinuities. They find that by searching for the best correlation obtained over a range of warpings better rejection of bad features is obtained, since correlation scores for good features will get worse to a certain extent due to viewpoint changes. They do not, however, attempt to estimate the orientation of the planes on which their patches implicitly lie: a search must always take place through possible warpings, which is computationally costly.

- Using line features is another possibility, although it has not yet been decided how best to incorporate these into the map-building system of Chapter 5. Line features would require new active strategies since it is not so clear where to fixate the camera when making measurements of them. Line features are now commonly used in structure from motion systems [83, 21, 103]. Having this capability would make a wider range of environmental objects usable as features, and perhaps lead to improvements in autonomous navigation capabilities since line features frequently form the boundaries of obstacles, such as the edges of a door frame.

4.2 Fixation

Fixation means directing a camera or cameras to point directly at a feature in the scene: the camera optic axes will pass through the feature, which will be imaged at the principal points. Fixation is one of the main tools of active vision: it means that image processing takes place always near the centre of images, where discrepancies are most easily dealt with — there is little danger of objects of interest being lost outside image boundaries. Many vision tasks can be simplified with simple fixated processing. A fixated camera acts as a "pointing stick" towards a feature. In addition, image processing near to the centre of an image reduces reliance on calibration parameters, as will be shown in Section 4.2.2.

In the work in this thesis, fixation is used whenever possible to make measurements of the positions of features.

4.2.1 Acquiring Features

Acquiring a new landmark consists of detecting a good patch in an image, then driving the active head to fixate the feature so that a measurement can be obtained. The feature detector of Section 4.1.1 is applied in the left image, and an epipolar line is generated in the right image: this is the image in the right camera of the line in the scene upon which the feature must lie, and it can be determined from the head model and knowledge of the head angles. A search for a match of the feature is carried out in the close vicinity of this

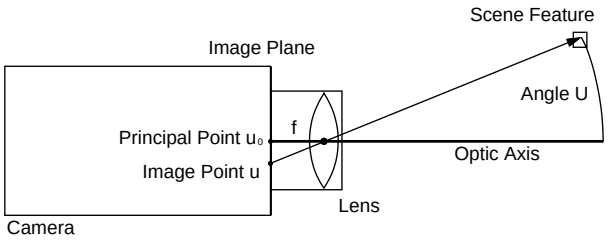


Figure 4.5: Relating pixel quantisation errors in the image plane to errors in perceived angle: this picture shows a two-dimensional view of camera imaging geometry.

line (8 pixels either side of the line). If the match is successful, the 3D location of the point relative to the head centre is calculated as in Section 3.3.3. The head angles necessary to fixate the feature are then determined, and the head is driven to this position. The feature should now be centred in both images: searches for the feature in circular regions near to the principal points of both are carried out. If the feature is found within a radius of 2 pixels of the principal point in both images, the fixation is successful; otherwise, its position and new head angles are re-calculated, and fixation is re-attempted. If fixation fails several times, the feature is abandoned: a mismatch is usually to blame.

4.2.2 The Accuracy of Fixated Measurements

In image measurements, we can expect to detect features to an accuracy of ± 1 pixel in normal circumstances. How does this translate into angular errors in fixation? Referring to Figure 4.5 which shows a simple view of imaging geometry in one plane, we can relate the angle to a scene feature U to its image position u via the equation:

$$\tan U = \frac{u - u_o}{f k_u},$$

where f is the focal length of the lens and k_u is the pixel density on the image plane measured in pixels m^{-1}. Differentiating this to determine how small changes δu in u relate to small changes δU in U gives:

$$\sec^2 U \delta U = \frac{\delta u}{f k_u}.$$

Now, when the camera is fixated on a feature, or close to fixation, then the angle a ray from that feature makes to the optic axis is very small: $U \approx 0$. Hence $sec^2 U \approx 1$ (very closely since $sec^2 U$ has a stationary point at $U = 0$).

$$\Rightarrow \delta U = \frac{\delta u}{f k_u}.$$

Since u is measured in pixels, when fixated to an accuracy of 1 pixel then $\delta u = 1$.

$$\Rightarrow \delta U = \frac{1}{f k_u} \ .$$

For the values of $f k_u$ and $f k_v$ in our cameras, this gives an angular error of:

$$\delta U \approx 0.006\text{rad} \approx 0.3^\circ \ .$$

Depth z (m)	$\gamma_R = -\gamma_L$ (radian)	Δx (m)	Δz (m)
0.2	0.6987	0.0014	0.0019
0.5	0.3241	0.0022	0.0070
1	0.1664	0.0044	0.026
2	0.08380	0.0086	0.10
3	0.05594	0.013	0.22
4	0.04198	0.017	0.40
5	0.03359	0.022	0.64
6	0.02799	0.026	0.92
7	0.02400	0.030	1.2
8	0.02100	0.034	1.6
9	0.01866	0.038	2.0
10	0.01680	0.042	2.6
15	0.01120	0.064	5.6
20	0.008400	0.084	10.0

Table 4.1: The uncertainties of the positions of fixation points measured by binocular cameras with inter-ocular spacing 0.338m and angular vergence uncertainties of 0.006rad.

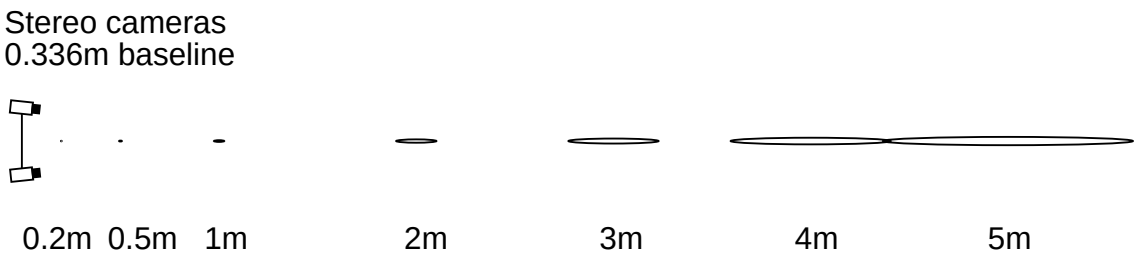


Figure 4.6: In this scale diagram ellipses show the uncertainties (1σ standard deviation) in recovered point positions from stereo fixation up to depths of 5m.

Compared to this, angular errors introduced by the active head, whose axes have repeatabilities with a maximum error of 0.005°, are negligible. Therefore, in all head measurements, an uncertainty of approximately 0.006rad is assigned to angular values obtained from head/image measurements. These uncertainties are directly incorporated into the localisation and map-building filter of Chapter 5.

Translating this into uncertainties in the positions of features estimated using fixated stereo measurements is revealing. Table 4.1 shows the depth (z coordinate) and transverse (x coordinate) uncertainty in estimated feature locations when measurements are made of features lying at varying distances directly in front of the robot. Figure 4.6 shows uncertainty ellipses to scale for the distances up to 5m. It is clear that while the transverse uncertainty remains small, growing only to 8cm at a distance of 20m, the depth uncertainty grows rapidly as the feature distance increases. This is because at large depths the two cameras are nearly parallel, and the vergence angles change very little for large distance changes: the small uncertainties in vergence measurements translate into large depth uncertainties. It will be seen later how this uncertainty distribution affects navigational capabilities.

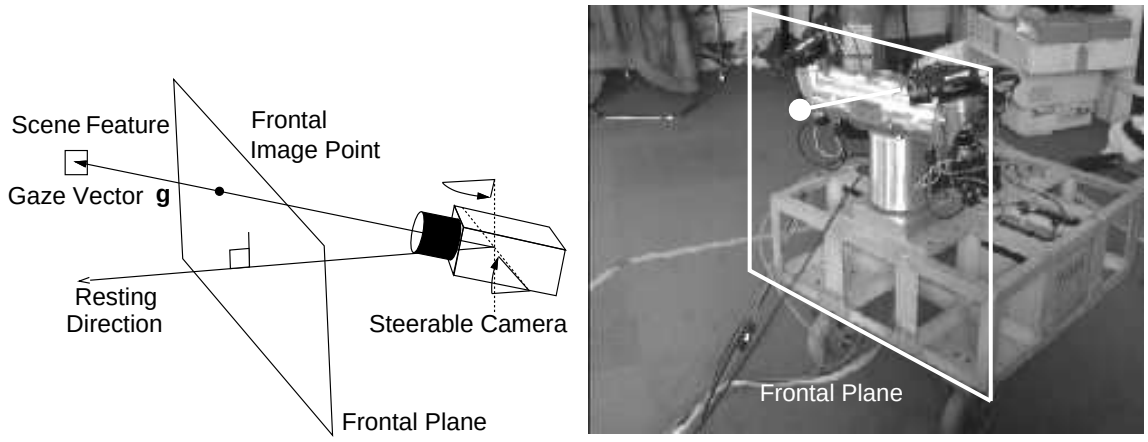


Figure 4.7: The frontal plane: (a) shows the mathematical concept of the plane and elevating/verging camera. (b) shows how the notional plane relates to our vehicle and active head.

4.3 An Example Use of Vision Tools: The Frontal Plane and Gazesphere

In the second part of this chapter, two ideas will be introduced which provide ways to visualise the field of view provided by an active camera, and using the methods developed in the first half of the chapter, implementation of an example of simple active-vision based navigation will be described.

The frontal plane and gazesphere concepts are most easily applied to a single active camera, and we will demonstrate their use in this case in an active algorithm which recovers single-step motions of the robot [24].

- The frontal plane [77] (also referred to as the virtual camera [79]) concept is best explained by Figure 4.7. A notional plane is imagined to lie at an arbitrary distance (usually taken as unity) in front of the active camera's centre of rotation, and perpendicular to its resting forward direction. When the active camera moves and fixates on a feature, its optic axis will intersect the plane at a point which can be referred to as the frontal plane image of the feature. Using the robot's left camera, once the vector $\mathbf{h}$ from the left camera optic centre to the feature has been determined and transformed into the vehicle frame $C0$, the frontal plane image point (X, Y) is trivially found from the equation:

$$\begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \propto \mathbf{h}^{C0} \ .$$

  Note that a small approximation is made here: that the centre of rotation of the active camera is the same as its optic centre. This is not quite true due to the offsets of the active head. However, the optic centre does not move much with rotations of

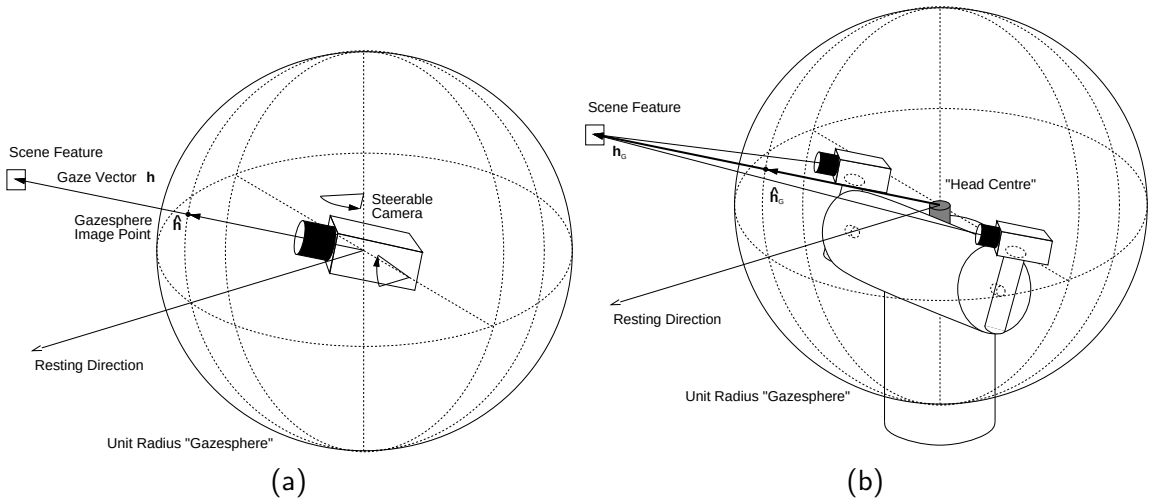


Figure 4.8: The notional gazesphere in place about (a) the centre of rotation of a single active camera, and (b) the head centre of a stereo camera platform.

the elevation and vergence axes (the same could not be said of movements of the pan axis, due to its relatively large baseline).

The combination of camera centre of rotation and frontal plane is mathematically equivalent to the optic centre and image plane of a physical passive camera. Therefore, any vision algorithms whose inputs are feature image positions for a passive camera can be used with frontal plane image positions instead. This makes a large number of methods immediately usable by a fixating active camera: although details of implementation may be very different from the passive case, the same mathematical principals can be applied. The notional frontal plane camera has a very wide field of view compared to its passive physical analogue due to the active camera's agility.

- The gazesphere is a very similar idea, and is depicted in Figure 4.8. Now a notional sphere is imagined surrounding the fixating camera, as in Figure 4.8(a). The optic axis intersects the sphere to form a gazesphere image of the fixated feature. The gazesphere image point $\hat{\mathbf{h}}^{C0}$ is calculated simply as the unit vector parallel to the scene vector $\mathbf{h}^{C0}$:

$$\hat{\mathbf{h}} = \frac{\mathbf{h}^{C0}}{|\mathbf{h}^{C0}|} \ .$$

  We now have a spherical projection of the world which is equivalent to the image acquired by a passive camera with a spherical image plane. Although this is a device rarely seen in practice due to the difficulties of making one, it would perhaps be the ideal way to make a camera since equal image areas always correspond to equal regions of solid angle in the scene. Various vision approaches (e.g. [30, 33]) have transformed planar image coordinates into a spherical form before processing, as this is a very natural way to represent imaging, and numerical advantages can be obtained as will be demonstrated in Section 4.3.3.

  The spherical representation is ideally suited to a fully mobile active camera, since

it deals naturally with views in all directions, including backwards: the frontal plane construction would have to be extended with another plane to deal with rear views. With our camera geometry, controlling just one camera via the elevation and vergence axis does not allow it to be directed backwards — however, we can also form the gazesphere construction around around the whole active head, as shown in Figure 4.8(b). The head centre now acts as the centre of the gazesphere. From a stereo view of a feature, the head centre vector $\mathbf{h}_G$ can be calculated, and the intersection of this with the sphere is the gazesphere image point. Using all its axes like this, the head is able to reach a field of view which is nearly fully panoramic.

In both of these constructions, the active camera is used as a projective pointing stick, fixation providing simple direction measurements to features which can be simply read from the head encoders. This means that both of the representations are effectively calibrated: considering the frontal plane especially, the exact relationship between image positions and scene angles is known. This makes it equivalent to a calibrated passive camera when it is used with passive algorithms.

In the following section we will describe the 8-point algorithm for structure from motion which was developed to determine the motion of a passive camera. Then, Sections 4.3.2 and 4.3.3 will demonstrate how the frontal plane and gazesphere constructions allow its implementation with an active camera to determine robot motion, and discuss the advantages this provides.

4.3.1 The 8-Point Algorithm for a Passive Camera

The actual version of the 8-point algorithm used is as described by Weng, Huang and Ahuja in [99]. Recent work by Hartley [38] has demonstrated that the 8-point algorithm in this form still compares well with the best iterative, non-linear methods for tackling two-view structure from motion problems.

Consider a camera moving in a scene as shown in Figure 4.9. In a certain discrete motion, its optic centre moves along a vector $\mathbf{t}$ in space from O_A to O_B, and the camera body is rotated by the matrix $\mathtt{R}^{\mathtt{AB}}$. Coordinate frames A and B are aligned with the camera in the initial and final positions respectively. $\mathbf{h}_A$ is the vector from O_A to a point P in the scene, and $\mathbf{h}_B$ is the vector from O_B to P.

We can deduce the simple spatial vector equation:

$$\mathbf{h}_A = \mathbf{h}_B + \mathbf{t}.$$

$\mathbf{h}_A$, $\mathbf{h}_B$ and $\mathbf{t}$ are coplanar: they define what is called the *epipolar plane*. Hence the scalar triple product of these vectors is equal to zero. We write:

$$\mathbf{h}_A \cdot (\mathbf{t} \times \mathbf{h}_B) = 0.$$

In the coordinates of frame A:

$$\begin{pmatrix} h^A_{Ax} & h^A_{Ay} & h^A_{Az} \end{pmatrix} \mathtt{E} \begin{pmatrix} h^B_{Bx} \\ h^B_{By} \\ h^B_{Bz} \end{pmatrix} = 0 \ , \tag{4.2}$$

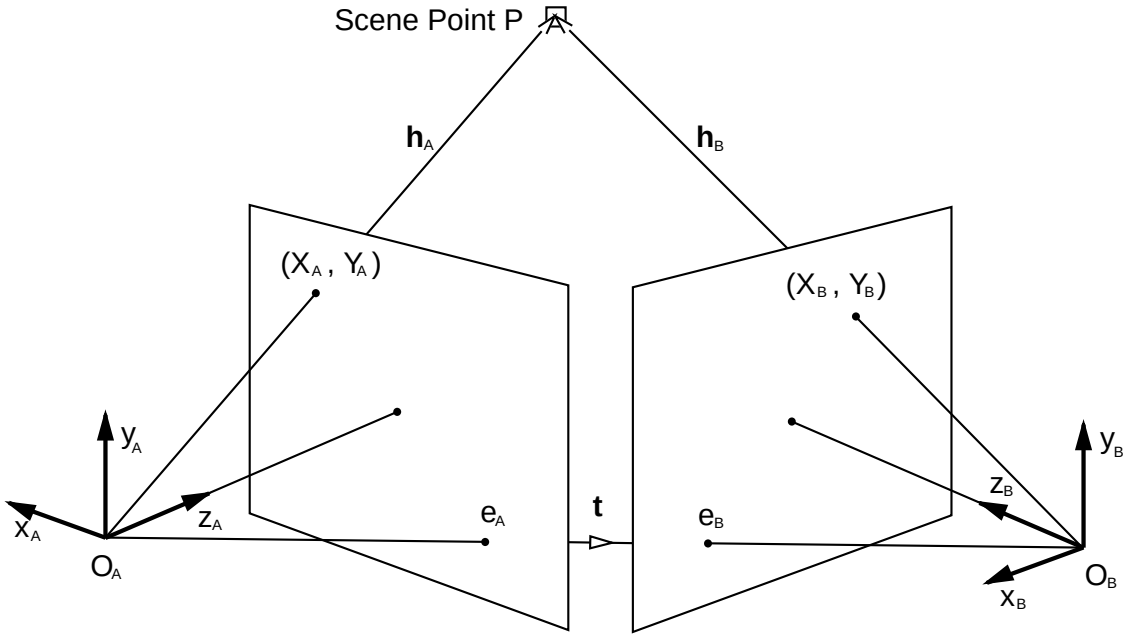


Figure 4.9: A scene point P is viewed by a camera in two successive positions. The camera moves along vector $\mathbf{t}$ and rotates according to matrix $\mathtt{R}^{\mathtt{AB}}$ between its initial and final positions. (X_A, Y_A) and (X_B, Y_B) are image points on ideal image planes at unit focal length.

where the essential matrix is $\mathtt{E} = [\mathtt{t}_\times]\mathtt{R}^{\mathtt{AB}}$, in which the antisymmetric matrix $[\mathtt{t}_\times]$ is made from the components of the translation vector:

$$[\mathtt{t}_\times] = \begin{bmatrix} 0 & -t_z^A & t_y^A \\ t_z^A & 0 & -t_x^A \\ -t_y^A & t_x^A & 0 \end{bmatrix}.$$

The essential matrix encodes the information about the rotation and translation between the two camera positions. The task is to use image information to determine $\mathtt{E}$ and then decompose it to recover $\mathtt{R}^{\mathtt{AB}}$ and $\mathbf{t}$.

If we know the ideal image coordinates (X_A, Y_A) and (X_B, Y_B) of the scene point P as seen from the two positions, each scene vector in Equation 4.2 can be replaced with the appropriate image coordinate vector to which it is equal up to a scale factor:

$$\begin{pmatrix} X_A & Y_A & 1 \end{pmatrix} \mathtt{E} \begin{pmatrix} X_B \\ Y_B \\ 1 \end{pmatrix} = 0. \tag{4.3}$$

If there are n points in the scene which can be seen from both camera positions, for each point i we can write:

$$\begin{pmatrix} X_{Ai} & Y_{Ai} & 1 \end{pmatrix} \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{21} & E_{22} & E_{23} \\ E_{31} & E_{32} & E_{33} \end{bmatrix} \begin{pmatrix} X_{Bi} \\ Y_{Bi} \\ 1 \end{pmatrix} = 0,$$

where now we have written $\mathtt{E}$ in component form. This is equivalent to the equation:

$$\begin{pmatrix} X_{Ai}X_{Bi} & X_{Ai}Y_{Bi} & X_{Ai} & Y_{Ai}X_{Bi} & Y_{Ai}Y_{Bi} & Y_{Ai} & X_{Bi} & Y_{Bi} & 1 \end{pmatrix} \mathbf{E}^\top = 0$$

where $\mathbf{E} = (\ E_{11} \quad E_{12} \quad E_{13} \quad E_{21} \quad E_{22} \quad E_{23} \quad E_{31} \quad E_{32} \quad E_{33}\ )^\top$. To compute the essential matrix, we therefore find the vector $\mathbf{E}$ which minimizes $||\mathtt{A}\mathbf{E}||$, where the rows of $\mathtt{A}$ are constructed from the image coordinates of each scene point i:

$$\mathtt{A} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_{Ai}X_{Bi} & X_{Ai}Y_{Bi} & X_{Ai} & Y_{Ai}X_{Bi} & Y_{Ai}Y_{Bi} & Y_{Ai} & X_{Bi} & Y_{Bi} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}. \tag{4.4}$$

$\mathbf{E}$ can be found as the unit eigenvector associated with the smallest eigenvalue of $\mathtt{A}^\top\mathtt{A}$ as long as $n \geq 8$ — at least eight point correspondences are needed, and the solution will be better with more.

The method by which $\mathtt{E}$ is decomposed into $\mathtt{R}^{\mathtt{AB}}$ and $\mathbf{t}$ is explained in detail in [99] but we will briefly outline the procedure, omitting certain sign checks: the unit vector $\hat{\mathbf{t}}$ is found by minimising $||\mathtt{E}^\top\hat{\mathbf{t}}||$. The minimisation can be achieved by singular value decomposition of $\mathtt{E}^\top$ into $\mathtt{E}^\top = \mathtt{U}\mathtt{W}\mathtt{V}^\top$. The unit vector will be the right singular vector $\mathbf{V}_i$ associated with the smallest singular value, which should be zero or close to zero. The rotation matrix $\mathtt{R}^{\mathtt{AB}}$ is determined by minimising $||{\mathtt{R}^{\mathtt{AB}}}^\top[-\mathbf{t}_\times] - \mathtt{E}^\top||$.

Once $\mathtt{R}$ and $\mathbf{t}$ have been calculated, the 3D scene positions or *structure* of all the n features matched between the two viewpoints can be found. This can be done simply by back-projecting rays from the camera in its two positions according to each feature's image coordinates and finding their intersection.

So a summary of what the 8-point algorithm does: as a passive camera moves on a trajectory through a scene, if between two positions on the trajectory we can match at least eight scene points in the image plane, we are able to calculate the interim rotation and direction of translation of the camera as well as the 3D scene positions of the reference features. Two points to note are:

- The rotation, translation and structure recovered are *Euclidean* only if the camera is *calibrated* (as the frontal plane representation is). If we do not know explicitly all the components of its camera calibration matrix $\mathtt{C}$ then these quantities can be determined only up to a projective ambiguity.
- The algorithm cannot recover the absolute scale of movements — it returns only a unit translation vector describing the *direction* of motion. The structure calculated has the same ambiguity.

4.3.2 The 8-Point Algorithm in The Frontal Plane

Method

The active implementation of the algorithm is straightforward using the ideas developed up to now. The frontal plane idea is used to create the analogue of a fixed wide-angle camera attached to the vehicle. By operating the 8-point algorithm on point correspondences in the frontal plane when the vehicle makes a discrete motion from an initial to a final position we can hence determine its rotation and translation.

- In the initial position the scene is examined for prominent features by the active camera in a wide field of view.
- Features are chosen by hand or automatically; for each one an image patch is saved and the active camera fixates it via the iterative search procedure described in Section 4.2.1.
- Once fixation is achieved, the head angles θ_e and θ_v from odometry are used to calculate the frontal plane image position of each feature.
- When a sufficient number of features has been identified from the initial position the vehicle is free to drive somewhere new. Generally, although only eight matches are required as a minimum, more features than this will be identified from the initial position to improve accuracy and allow for potential matching failures.
- After moving, as many features as possible are re-found via correlation matching and re-fixated iteratively.
- The head angles at fixation are used to calculate new frontal plane coordinates for the features.
- Finally the set of successfully matched frontal plane coordinates is passed to the 8-point algorithm which calculates the rotation and direction of translation of the vehicle.

Experiments

Several experiments with known motions of the vehicle were performed to verify the correct operation of the system and investigate its accuracy. Figure 4.10 shows the motions which were used.

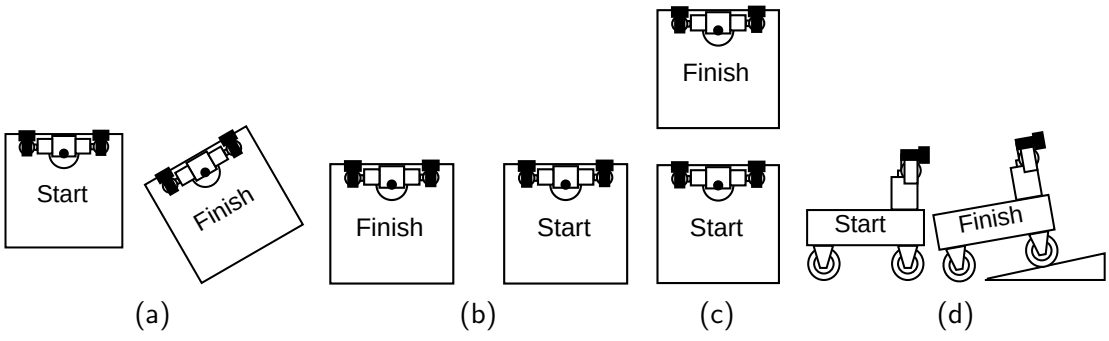


Figure 4.10: The vehicle motions used in Experiments 1 (a), 2 (b,c) and 3 (d).

Experiment 1: translation and rotation in the ground plane. In the first experiment the vehicle was moved on the ground plane as shown in Figure 4.10(a) with translation towards its right and backwards, and a rotation to the left measured as 30°. The direction of translation was not measured in this case since due to the rotation of the vehicle it was

First Position				Second Position			
Joint Angles		Frontal Plane		Joint Angles		Frontal Plane	
$\theta_e(^\circ)$	$\theta_v(^\circ)$	x	y	$\theta'_e(^\circ)$	$\theta'_v(^\circ)$	x'	y'
2.35	-13.11	-0.233	0.041	2.69	-37.42	-0.766	0.047
13.33	-14.20	-0.260	0.237	16.38	-37.01	-0.786	0.294
23.94	-20.31	-0.405	0.444	31.59	-41.64	-1.044	0.615
33.30	-35.52	-0.854	0.657	52.09	-53.14	-2.171	1.284
24.18	-41.24	-0.961	0.449	43.53	-61.48	-2.538	0.950
5.71	-44.10	-0.974	0.100	11.91	-68.24	-2.561	0.211
40.20	30.10	0.759	0.845	32.33	11.00	0.230	0.633
36.50	21.78	0.497	0.740	32.82	1.30	0.027	0.645
5.99	3.41	0.060	0.105	6.22	-21.04	-0.387	0.109
18.26	29.92	0.606	0.330	15.32	6.49	0.118	0.274

Table 4.2: Joint angles and frontal plane coordinates to fixate on features in successive positions of the vehicle in Experiment 1.

difficult to tell how exactly the centre of rotation of the left camera had moved. The head angles obtained for fixation on ten scene features from each view and the frontal plane coordinates derived from them are displayed in Table 4.2.

The expected or *veridical* rotation matrix $\mathtt{R}_v$ calculated for a precise 30° is compared below with the measured matrix $\mathtt{R}_m$ obtained from the 8-point algorithm:

$$\mathtt{R}_v = \begin{bmatrix} 0.867 & 0.000 & 0.500 \\ 0.000 & 1.000 & 0.000 \\ -0.500 & 0.000 & 0.867 \end{bmatrix} \quad \mathtt{R}_m = \begin{bmatrix} 0.871 & -0.011 & 0.490 \\ 0.014 & 1.000 & -0.003 \\ -0.490 & 0.010 & 0.872 \end{bmatrix}.$$

As can be seen, the agreement is very good, and the rotation has been accurately recovered. The measured direction of translation denoted by unit vector $\hat{\mathbf{t}}_m$ was:

$$\hat{\mathbf{t}}_m = \begin{pmatrix} -0.855 \\ -0.142 \\ -0.498 \end{pmatrix}.$$

This recovered translation is not quite so satisfying, there being a component -0.142 in the y direction corresponding to an apparent downward motion of the vehicle which was not present. The reason for this is that the actual movement of the left camera in the xz plane parallel to the ground was very small in this particular motion and so the -0.142 in the unit vector is actually a tiny distance when the unit vector is scaled up. In the next two experiments, with significant horizontal vehicle translations, the recovered translation directions are much more convincing.

Experiment 2: translation only. Two motions in the ground plane with no rotation were investigated. The vehicle was first moved in a simple sideways translation to the left as in Figure 4.10(b) and then forwards as in Figure 4.10(c). The absolute distance of movement in both cases was about 1 metre. The veridical and measured motions in the

first motion, where 11 feature correspondences were obtained, were:

$$\hat{\mathbf{t}}_v = \begin{pmatrix} 1.000 \\ 0.000 \\ 0.000 \end{pmatrix} \quad , \quad \hat{\mathbf{t}}_m = \begin{pmatrix} 0.999 \\ -0.003 \\ 0.051 \end{pmatrix}$$

$$\mathtt{R}_v = \begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix} \quad , \quad \mathtt{R}_m = \begin{bmatrix} 1.000 & -0.004 & -0.009 \\ 0.004 & 1.000 & 0.006 \\ 0.009 & -0.006 & 1.000 \end{bmatrix}.$$

For the second translation, the veridical and measured motion estimated for 12 correspondences were:

$$\hat{\mathbf{t}}_v = \begin{pmatrix} 0.000 \\ 0.000 \\ 1.000 \end{pmatrix} \quad , \quad \hat{\mathbf{t}}_m = \begin{pmatrix} -0.016 \\ -0.003 \\ 1.000 \end{pmatrix}$$

$$\mathtt{R}_v = \begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix} \quad , \quad \mathtt{R}_m = \begin{bmatrix} 1.000 & 0.020 & -0.042 \\ -0.020 & 1.000 & -0.009 \\ 0.041 & 0.010 & 1.000 \end{bmatrix}.$$

It can be seen that for both of these motions the measured quantities are very close to the expected axial unit translation vectors and identity rotation matrices.

Experiment 3: rotation and translation out of the ground plane. To demonstrate that recoverable motions are not confined to the ground plane, the front of the vehicle was raised up so that the vehicle's base made an angle of 10° with the ground as in Figure 4.10(d). Using 19 matched points, the recovered motion was:

$$\hat{\mathbf{t}}_m = \begin{pmatrix} 0.109 \\ 0.209 \\ 0.972 \end{pmatrix}$$

$$\mathtt{R}_v = \begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.000 & 0.985 & 0.174 \\ 0.000 & -0.174 & 0.985 \end{bmatrix} \quad , \quad \mathtt{R}_m = \begin{bmatrix} 1.000 & 0.005 & 0.001 \\ -0.005 & 0.983 & 0.183 \\ -0.000 & -0.183 & 0.983 \end{bmatrix}.$$

We have compared the measured rotation with the matrix calculated for a precise 10° rotation in the vertical yz plane and the agreement is again good.

4.3.3 The 8-Point Algorithm in the Gazesphere

The 8-point algorithm can be adapted slightly to work with spherical image coordinates, and this presents two advantages:

- Features can now lie in any direction relative to the centre of rotation, not just forwards. This means that matching can be achieved over any rotations of the robot.
- The improved numerical conditioning of the spherical representation means that the accuracy of motions recovered from the gazesphere is slightly better than from the frontal plane.

	Fixation Angles From Vehicle Positions (radians)							
Feature	A		B		C		D	
Coordinates	θ_e	θ_v	θ_e	θ_v	θ_e	θ_v	θ_e	θ_v
(3, 2, 5)	0.380506	0.508267	0.380506	0.748327	0.588003	0.693981	0.755969	1.038493
(6, 0, 8)	0.000000	0.643501	0.000000	0.785398	0.000000	0.785398	0.000000	1.343997
(4, 4, 10)	0.380506	0.355603	0.380506	0.508267	0.463648	0.420534	0.679674	0.965252
(2, -1, 8)	-0.124355	0.243161	-0.124355	0.460554	-0.165149	0.317663	-0.199347	0.900036
(-1, 0, 12)	0.000000	-0.083141	0.000000	0.083141	0.000000	-0.099669	0.000000	0.620249
(-2, 2, 10)	0.197396	-0.193658	0.197396	0.000000	0.244979	-0.237941	0.214233	0.484350
(-4, -2, 11)	-0.179853	-0.343581	-0.179853	-0.177013	-0.218669	-0.409352	-0.174969	0.353742
(-5, 3, 8)	0.358771	-0.529470	0.358771	-0.337675	0.463648	-0.640522	0.294249	0.135875
(-4, -2, 5)	-0.380506	-0.638865	-0.380506	-0.355603	-0.588003	-0.837215	-0.275643	-0.000000
(-7, 1, 5)	0.197396	-0.941243	0.197396	-0.775594	0.321751	-1.146485	0.108360	-0.225513

Table 4.3: The head angles for precise fixation on the scene points from the four vehicle vehicle positions.

To use the 8-point algorithm with spherical coordinates, it is necessary just to replace the image vectors of Equation 4.3 with gazesphere image vectors to give the constraint provided by each feature match as:

$$\begin{pmatrix} \hat{h}_{Ax}^{C0} & \hat{h}_{Ay}^{C0} & \hat{h}_{Az}^{C0} \end{pmatrix} \mathtt{E} \begin{pmatrix} \hat{h}_{Bx}^{C0} \\ \hat{h}_{By}^{C0} \\ \hat{h}_{Bz}^{C0} \end{pmatrix} = 0 \ . \tag{4.5}$$

Hence the rows of matrix $\mathtt{A}$ as in Equation 4.4 are now constructed from spherical coordinate components:

$$\mathtt{A} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hat{h}_{Ax_i}^{C0}\hat{h}_{Bx_i}^{C0} & \hat{h}_{Ax_i}^{C0}\hat{h}_{By_i}^{C0} & \hat{h}_{Ax_i}^{C0}\hat{h}_{Bz_i}^{C0} & \hat{h}_{Ay_i}^{C0}\hat{h}_{Bx_i}^{C0} & \hat{h}_{Ay_i}^{C0}\hat{h}_{By_i}^{C0} & \hat{h}_{Ay_i}^{C0}\hat{h}_{Bz_i}^{C0} & \hat{h}_{Az_i}^{C0}\hat{h}_{Bx_i}^{C0} & \hat{h}_{Az_i}^{C0}\hat{h}_{By_i}^{C0} & \hat{h}_{Az_i}^{C0}\hat{h}_{Bz_i}^{C0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{4.6}$$

We can then go on as before to find $\mathtt{E}$ and decompose it to recover $\mathtt{R}$ and $\hat{\mathbf{t}}$.

Spherical Coordinates Versus Frontal Plane Coordinates: A Comparison Using Simulated Data

A set of tests with fabricated data was performed to verify the numerical advantages of using spherical coordinates with the 8-point algorithm.

A set of ten 3D feature points in space was invented and the vehicle imagined to move on the ground plane between four positions amongst them (chosen such all the points were visible at all times). Figure 4.11 shows the situation from a top-down viewpoint (the feature points also had heights as denoted in their coordinates). From each of the vehicle positions A, B, C and D, the head angles necessary for fixation on all of the features were calculated. These are shown in Table 4.3.

Gaussian noise of magnitude equal to the noise expected in angular head measurements (see Section 4.2.2) was added to the angles, and they were converted into both frontal plane and spherical coordinates as explained earlier. Matched sets of these coordinates for the motions A $\rightarrow$ B, A $\rightarrow$ C and A $\rightarrow$ D were then fed to the 8-point algorithm to see how well the motions were recovered with the different coordinate types.

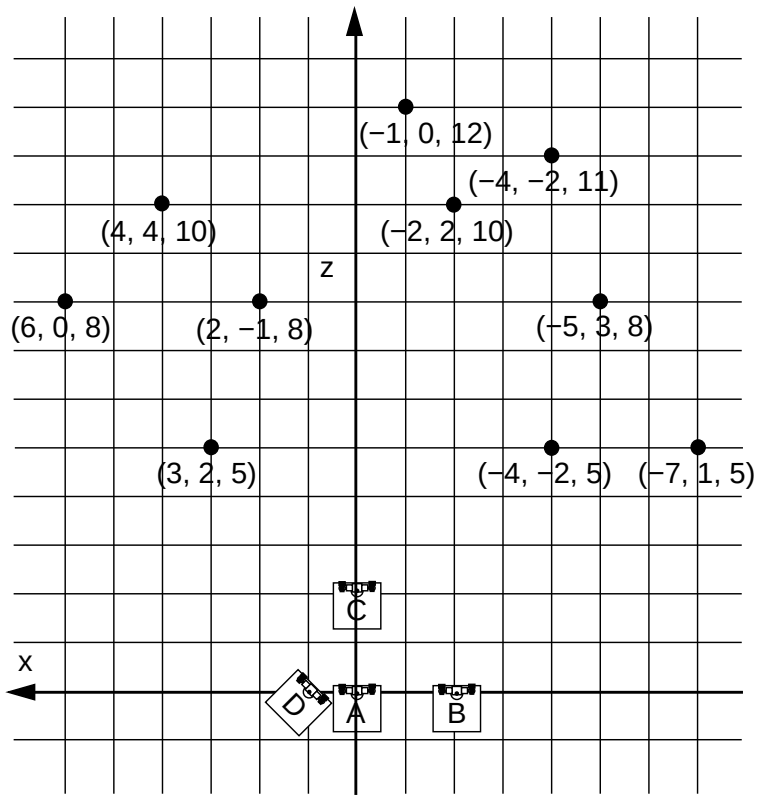


Figure 4.11: An above-head view of the vehicle and feature positions used in the numerical comparison of spherical and frontal plane coordinates.

The true motions of the ideal vehicle, described as a unit translation vector $\hat{\mathbf{t}}$ and a rotation matrix $\mathtt{R}$ were:

$$A \to B: \quad \hat{\mathbf{t}} = \begin{pmatrix} -1.000000 \\ 0.000000 \\ 0.000000 \end{pmatrix} \quad \mathtt{R} = \begin{bmatrix} 1.000000 & 0.000000 & 0.000000 \\ 0.000000 & 1.000000 & 0.000000 \\ 0.000000 & 0.000000 & 1.000000 \end{bmatrix}$$

$$A \to C: \quad \hat{\mathbf{t}} = \begin{pmatrix} 0.000000 \\ 0.000000 \\ 1.000000 \end{pmatrix} \quad \mathtt{R} = \begin{bmatrix} 1.000000 & 0.000000 & 0.000000 \\ 0.000000 & 1.000000 & 0.000000 \\ 0.000000 & 0.000000 & 1.000000 \end{bmatrix}$$

$$A \to D: \quad \hat{\mathbf{t}} = \begin{pmatrix} 1.000000 \\ 0.000000 \\ 0.000000 \end{pmatrix} \quad \mathtt{R} = \begin{bmatrix} 0.707107 & 0.000000 & -0.707107 \\ 0.000000 & 1.000000 & 0.000000 \\ 0.707107 & 0.000000 & 0.707107 \end{bmatrix}$$

For the noisy data and the *frontal plane* coordinate system, the recovered motions were:

$$A \to B: \quad \hat{\mathbf{t}} = \begin{pmatrix} -0.999862 \\ 0.008075 \\ 0.014543 \end{pmatrix} \quad \mathtt{R} = \begin{bmatrix} 0.999944 & -0.000866 & -0.010591 \\ 0.000946 & 0.999972 & 0.007467 \\ 0.010584 & -0.007476 & 0.999916 \end{bmatrix}$$

$$A \to C: \quad \hat{\mathbf{t}} = \begin{pmatrix} -0.023559 \\ -0.009083 \\ 0.999681 \end{pmatrix} \quad \mathtt{R} = \begin{bmatrix} 0.999971 & 0.000830 & 0.007504 \\ -0.000860 & 0.999992 & 0.003902 \\ -0.007501 & -0.003909 & 0.999964 \end{bmatrix}$$

$$A \to D: \quad \hat{\mathbf{t}} = \begin{pmatrix} 0.997672 \\ -0.038149 \\ -0.056524 \end{pmatrix} \quad \mathtt{R} = \begin{bmatrix} 0.699795 & -0.003049 & -0.714337 \\ 0.017624 & 0.999760 & 0.012998 \\ 0.714126 & -0.021685 & 0.699681 \end{bmatrix}$$

The motions recovered using *spherical* coordinates were:

$$A \to B: \quad \hat{\mathbf{t}} = \begin{pmatrix} -0.999819 \\ 0.006472 \\ 0.017901 \end{pmatrix} \quad \mathtt{R} = \begin{bmatrix} 0.999964 & -0.000233 & -0.008481 \\ 0.000277 & 0.999987 & 0.005137 \\ 0.008480 & -0.005139 & 0.999951 \end{bmatrix}$$

$$A \to C: \quad \hat{\mathbf{t}} = \begin{pmatrix} -0.022316 \\ -0.008430 \\ 0.999715 \end{pmatrix} \quad \mathtt{R} = \begin{bmatrix} 0.999974 & 0.000638 & 0.007145 \\ -0.000664 & 0.999993 & 0.003719 \\ -0.007143 & -0.003724 & 0.999968 \end{bmatrix}$$

$$A \to D: \quad \hat{\mathbf{t}} = \begin{pmatrix} 0.998009 \\ -0.033777 \\ -0.053257 \end{pmatrix} \quad \mathtt{R} = \begin{bmatrix} 0.700428 & -0.003897 & -0.713713 \\ 0.015756 & 0.999826 & 0.010004 \\ 0.713549 & -0.018252 & 0.700367 \end{bmatrix}$$

It can be seen that the spherical coordinates perform consistently slightly better than the frontal plane coordinates with exactly the same input data, being closer to the perfect values in nearly all cases. The reason for this is the better numerical conditioning of the spherical coordinates which makes them more suitable for constructing the rows of the matrix `A` which is decomposed to find `E`. The spherical coordinate values all lie in the range $0 \to 1$ whereas the frontal plane coordinates can be unlimited in size and tend to ∞ for features at large angles. Small angular errors will lead to large frontal plane position errors at these positions. With spherical coordinates, a certain angular error will lead to the same change in coordinate values whatever the position of the feature — the noise in the system, whose form is not specified in the 8-point algorithm, is dealt with better in this case.

4.4 Conclusion

This chapter has presented the vision tools for dealing with features, and shown how the frontal plane and gazesphere concepts can be used to implement a previously developed navigation algorithm. As is stands, however, the 8-point algorithm is not much use for extended periods of vehicle navigation, considering as it does only single motions. A much more difficult problem is constructing a system which can estimate compound motions from the same kind of feature measurements. It is also desirable to be able to make use of whatever number of measurements is available: the 8-point algorithm is unsuitable for practical active navigation since the robot must fixate in turn upon at least eight features to determine its position — a time-consuming process. A measurement of just one feature should be sufficient to provide at least some information on motion.

In the next chapter, we will present the main part of the map-building and localisation algorithm developed to solve these problems for long-term robot navigation.

5

Map Building and Sequential Localisation

Navigating in unknown surroundings inherently couples the processes of building a map of the area and calculating the robot's location relative to that map. In the previous chapter, an active implementation of a structure from motion algorithm was demonstrated which allows the calculation of a single motion of the robot. This method, however, has no natural extension to compound motions, where it is desirable to calculate the robot's location sequentially relative to a continuously updated map of features. It is necessary to consider the uncertainties introduced into the map by potential noise sources in the robot motion and measurement process.

In this chapter we will describe the map-building and localisation algorithm which we believe is key to a navigation system using active vision. The algorithm relies on the familiar Kalman Filter, but the particular way it has been used has been designed around the requirements of our robot navigation system. These requirements are that in previously unknown environments, maps of features can be automatically generated which are useful and efficient, in terms of providing landmarks which are easy and relevant to refer to, and consistent and long-lasting, meaning that they can be used for long periods of navigation in the same area. It is also a requirement that the localisation method is applicable to real robot tasks, where particular goals must be achieved, whether specified by a human operator or generated by the robot itself.

5.1 Kalman Filtering

The Kalman Filter is a general statistical tool for the analysis of time-varying physical systems in the presence of noise. A system is modelled by a state vector $\mathbf{x}$ which has entries

for each of the quantities of interest. The passage of time is divided into small intervals Δt, and knowledge of the expected evolution of the state vector is encapsulated in the state transition function $\mathbf{f}$.

The filter permits continuous and efficient estimation of $\mathbf{x}$ as the system evolves, incorporating the information provided by any measurements $\mathbf{z}$ which are made of quantities depending on the state. The current state estimate is stored in the vector $\hat{\mathbf{x}}$, and the covariance matrix $\mathtt{P}$, which is square and symmetric with dimension of the number of elements in $\mathbf{x}$, represents the uncertainty in $\hat{\mathbf{x}}$. If the dependence of both $\mathbf{f}$ and the measurement function $\mathbf{h}$ on $\mathbf{x}$ is linear, and the statistical distribution of noise in the state transition and measurements is Gaussian, then the solution produced is optimal.

5.1.1 The Extended Kalman Filter

The Extended Kalman Filter (EKF) is a simple extension of the Kalman Filter to cope with systems whose state transition and measurement functions are non-linear, or whose noise distributions are non-Gaussian. This is true of most physical systems to which the filter has been applied, and it acts only as an approximation in these cases, where the downside of its efficiency is oversimplification of the mathematical forms of the functions and distributions involved. Usually the results are quite adequate, but in some recent vision applications, such as tracking the outlines of rapidly moving objects against cluttered backgrounds, the Kalman Filter has been found to fail (in this case due to its inability to use multi-modal distributions to represent multiple hypotheses about the location of an object) and replacement algorithms have been developed [45]. In our system, the EKF has been found to work well, although it has been important to be careful with some aspects of formulation, as will be explained later.

Prediction

In a step of time during which no measurements are made, our estimate of the state of a system changes according to the state transition function $\mathbf{f}$ describing the dynamics. The covariance matrix changes reflecting the increase in uncertainty in the state due to noise $\mathtt{Q}$ in the state transition (due to random effects or factors which are not accounted for in the dynamic model). Both $\mathbf{f}$ and $\mathtt{Q}$ depend on $\mathbf{u}$, the current vehicle control vector specifying demanded velocity and steering angle. The label k denotes an incrementing time step.

$$\hat{\mathbf{x}}(k+1|k) = \mathbf{f}(\hat{\mathbf{x}}(k|k), \mathbf{u}(k)) \ , \tag{5.1}$$

$$\mathtt{P}(k+1|k) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(k|k)\mathtt{P}(k|k)\frac{\partial \mathbf{f}}{\partial \mathbf{x}}^{\top}(k|k) + \mathtt{Q}(k) \ . \tag{5.2}$$

Update

When a measurement is made, the state estimate improves with the new information and the uncertainty represented by $\mathtt{P}$ will reduce. ν is the difference between the actual measurement $\mathbf{z}$ and the prediction $\mathbf{h}$ calculated from the current state, and is called the innovation. $\mathtt{R}$ is the covariance matrix of the noise in the measurement. The update equations for state and covariance are:

$$\hat{\mathbf{x}}(k+1|k+1) = \hat{\mathbf{x}}(k+1|k) + \mathtt{W}(k+1)\nu(k+1) \ , \tag{5.3}$$

$$\mathtt{P}(k+1|k+1) \quad = \quad \mathtt{P}(k+1|k) - \mathtt{W}(k+1)\mathtt{S}(k+1)\mathtt{W}^{\top}(k+1) \ , \tag{5.4}$$

where the innovation is

$$\nu(k+1) = \mathbf{z}(k+1) - \mathbf{h}(\hat{\mathbf{x}}(k+1|k)) \ , \tag{5.5}$$

$\mathtt{W}$, the Kalman gain, is

$$\mathtt{W}(k+1) = \mathtt{P}(k+1|k)\frac{\partial \mathbf{h}}{\partial \mathbf{x}}^{\top}(k|k)\mathtt{S}^{-1}(k+1) \ , \tag{5.6}$$

and $\mathtt{S}$, the innovation covariance is

$$\mathtt{S}(k+1) = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}(k|k)\mathtt{P}(k+1|k)\frac{\partial \mathbf{h}}{\partial \mathbf{x}}^{\top}(k|k) + \mathtt{R}(k+1) \ . \tag{5.7}$$

The innovation covariance represents the uncertainty in ν, the amount by which a true measurement differs from its predicted value.

5.2 Simultaneous Map-Building and Localisation

5.2.1 Frames of Reference and the Propagation of Uncertainty

In a map-building and localisation system, the quantities of interest are the coordinates of map features and the location of the robot. It is these which will form the state vector of the filter.

When deciding upon the exact form of the filter, the issue of coordinate frames and their significance was considered in detail. Consider navigation around a restricted area such as a room: certainly there is no need to know about position or orientation relative the world as a whole, but just about the relative location of certain features of the room (walls, doors, etc.). This initially suggested a completely robot-centred approach to navigation: the positions of these features would be estimated in a robot-centred coordinate frame at all times, and the robot's location would not be explicitly described. A forward motion of the robot would appear simply as a backward motion of the features. This approach is attractive because it seems to encapsulate all the information that is important for navigation in initially unknown surroundings. However, such an approach cannot explicitly answer questions such as "how far has the robot moved between points A and B on its trajectory?", or even enable the robot to perform a task as simple as returning to its starting position after carrying out a particular motion. These capabilities are essential in goal-directed performance.

Therefore, the approach chosen explicitly estimates the robot and feature positions relative to a world frame while maintaining covariances between all the estimates. This provides a way to answer these questions while retaining all the functionality of the robot-centred method, since it is possible at any stage to re-zero the coordinate frame to the robot, as we will show in Section 5.3.8. The extra information held in the explicit robot state and its covariances with the feature estimates codes the registration information between an arbitrary world coordinate frame or map and the structure of the feature map the robot has built itself. This representation allows for the inclusion of prior information given to the robot about the world, as will be shown in Chapter 7.

Autonomous map-building is a process which must be carefully undertaken. A map which is made by a traveller or robot who does not have some external measure of ego-motion is fundamentally limited in its accuracy. The problem is caused by the compound errors of successive measurements. Consider, for example, a human given the task of drawing a very long, straight line on the ground, but equipped with only a 30cm ruler, and unable to use any external references such as a compass or the bearing of the sun. The first few metres would be easy, since it would be possible to look back to the start of the line when aligning the ruler to draw a new section. Once this had gone out of view, though, only the recently drawn nearby segment would be available for reference. Any small error in the alignment of this segment would lead to a misalignment of new additions. At a large distance from the starting point, the cumulative uncertainty will be great, and it will be impossible to say with any certainty whether the parts of line currently being drawn were parallel to the original direction. Changing the measurement process could improve matters: if, for instance, flags could be placed at regular intervals along the line which were visible from a long way, then correct alignment could be better achieved over longer distances. However, eventually the original flags would disappear from view and errors would accumulate — just at a slower rate than before.

Something similar will happen in a robot map-building system, where at a certain time measurements can be made of only a certain set of features which are visible from the current position — probably these will in general be those that are nearby, but there are usually other criteria as discussed in Section 4.1.2, such as occlusion or maximum viewing angle. It will be possible to be confident about the robot's position relative to the features which can currently be seen, but decreasingly so as features which have been measured in the more distant past are considered. Should the map-building algorithm chosen accurately reflect this if the maps generated are to be consistent and useful for extended periods? We argue yes, and show how this can be achieved by implementing localisation and map-building using a single Kalman Filter, where a large state vector contains all the quantities of interest (the robot and feature positions) and a large covariance matrix stores the related uncertainties of all of these.

In the majority of previous approaches to robot navigation and sequential structure from motion, (e.g. [36, 10, 8, 27]), separate filters have been used for the robot position and that of each of the features (or further simplified schemes, where, for instance, the robot state is not estimated explicitly with a covariance but is calculated by a least squares fit of the most recently measured features to those in the map). This approach is certainly attractive due to its reduction of the computational complexity of building and updating a map. Each particular estimated state vector — the small number of coordinates describing the location of the robot or one feature — is stored along with the small covariance matrix describing the uncertainty in that estimate. What are not stored, however, are the covariance cross-terms relating the estimates of the coordinates of different features or the robot. It has been argued previously [91, 92, 93] that this is not sufficient for fully effective map-building, although it has appeared to be effective in many demonstrations. What these demonstrations have failed to show, however, is the production of maps that are useful for any extended periods of navigation and the performance of robot tasks. Maps built in this way without full consideration of the uncertainties involved are prone to inconsistency. A familiar state of affairs is the observation that the trajectory calculated by a robot which has moved in a looped path does not join up at the ends — this is reported in work by both Harris and

Pike [36] and Bouget and Perona [10]. An experiment will be reported in Section 5.4.4 later in this chapter which demonstrates how the full covariance approach greatly improves on this.

Some previous authors have tackled the problem of producing real-time implementations of map-building using the full covariance matrix. McLauchlan's Variable State-Dimension Filter [63, 64] ingeniously reduces the complexity of updates by operating on the inverse of the covariance matrix (often called the information matrix), which in certain problems has a simple block-diagonal form. This is indeed the case with map-building in cases where there is no prior information about the motion of the observer — reconstruction of scenes from arbitrary video sequences for instance. It is not yet clear whether additional information such as robot odometry data can be incorporated into this simple form. Csorba [23] has presented an efficient algorithm which reduces the complexity of full covariance updates while reducing accuracy slightly to within a bounded range.

Our approach to the potential problem of the computational complexity of maintaining a large coupled filter is described in Chapter 6, where an algorithm is described which can reduce the amount of processing to be performed when making repeated measurements of just one feature in the map to a small, constant amount, by postponing the full, but exact, update of the whole state until the robot has stopped moving and has some unutilised processing time.

5.2.2 Map-Building in Nature

It is interesting to ponder the way that humans and animals deal with the map-building problem. It certainly does not seem that we store the equivalent of a huge covariance matrix relating the uncertainties in our estimates of the positions of all the features in a certain area. Within one part of that area, for example on a street in the middle of a large familiar city, it might be difficult to point in the direction of another street on the other side of the city, or to estimate its distance, but very easy to find the way there if necessary — from the current position, some local landmarks would enable the first street along the journey to be found, which would lead to the next, and so on to the final destination. Of course, a city is a simplified environment because generally paths through it are limited to the grid of streets; but humans make great use of the local relations between landmarks — it is only necessary to be able to find the few nearby points of interest from the current position. Canonical paths [69] are standard choices of route which will ease navigation, and these have been noted in many situations. Human navigators prior to the latter part of the eighteenth century used to "sail the parallels" [32], meaning that voyages were composed of straightforward north/south or east/west sections where navigation was simpler, rather than the more direct alternatives. We are also reminded of Leonard's suggestion that in robot navigation map contact [52] is often more important than other criteria, such as short distance, when planning a path.

The relationship between the positions of features is exactly what is monitored by the large covariance matrix in our map-building algorithm, but perhaps more information is being stored than is truly necessary. Full consideration of the relationship between features in a local area certainly seems worth keeping, but is it really important to know individually about the relationship between each of these features and a particular distant one, as is currently the case? The sort of environment is relevant: in open country, where there

are few landmarks, it is more important to know about how each relates to the others, since it may be necessary to "strike out" over a long distance between them without being able to refer to others along the way. For a robot operating in a relatively plain indoor environment, the situation is similar, since the robot will be aware of only a sparse map of interesting features — though in future systems, use of more advanced representations and feature types such as lines and planes might give the robot a human's instinctive grasp of the geometry of a room and his or her position in it.

Our system could become more like the apparent human one then by selectively storing feature relationships. In an indoor environment, they could be grouped into rooms for instance, and adjoining rooms coupled in some way. The covariance matrix would become more diagonal (if the features are listed in it in room order).

5.3 The Map-Building and Localisation Algorithm

5.3.1 The State Vector and its Covariance

Current estimates of the locations of the robot and the scene features which are known about are stored in the system state vector $\hat{\mathbf{x}}$, and the uncertainty of the estimates in the covariance matrix $\mathtt{P}$. These are partitioned as follows:

$$\hat{\mathbf{x}} = \begin{pmatrix} \hat{\mathbf{x}}_v \\ \hat{\mathbf{y}}_1 \\ \hat{\mathbf{y}}_2 \\ \vdots \end{pmatrix} \quad , \quad \mathtt{P} = \begin{bmatrix} \mathtt{P}_{xx} & \mathtt{P}_{xy_1} & \mathtt{P}_{xy_2} & \dots \\ \mathtt{P}_{y_1x} & \mathtt{P}_{y_1y_1} & \mathtt{P}_{y_1y_2} & \dots \\ \mathtt{P}_{y_2x} & \mathtt{P}_{y_2y_1} & \mathtt{P}_{y_2y_2} & \dots \\ \vdots & \vdots & \vdots & \end{bmatrix} . \tag{5.8}$$

$\hat{\mathbf{x}}$ has $3(n+1)$ elements, where n is the number of known features. $\mathtt{P}$ is symmetric, with size $3(n+1) \times 3(n+1)$. $\hat{\mathbf{x}}$ and $\mathtt{P}$ will change in dimension as features are added or deleted from the map. $\hat{\mathbf{x}}_v$ is the robot position estimate, and $\hat{\mathbf{y}}_i$ the estimated 3D location of the ith feature:

$$\hat{\mathbf{x}}_v = \begin{pmatrix} \hat{z} \\ \hat{x} \\ \hat{\phi} \end{pmatrix} \quad , \quad \hat{\mathbf{y}}_i = \begin{pmatrix} \hat{X}_i \\ \hat{Y}_i \\ \hat{Z}_i \end{pmatrix} .$$

These coordinates are all in the world frame W, but the frame superscript is dropped for clarity.

5.3.2 Filter Initialisation

The usual way to initialise the system is with no knowledge of any scene features, and with the world coordinate frame defined to be aligned with the robot's starting position. The robot's coordinates are $\hat{z} = 0$, $\hat{x} = 0$, $\hat{\phi} = 0$ by definition — and since these are known exactly, the starting covariance matrix has all entries equal to zero:

$$\hat{\mathbf{x}} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad , \quad \mathtt{P} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} . \tag{5.9}$$

5.3.3 Moving and Making Predictions

The robot's motion is discretised into steps of time interval Δt, with an incrementing label k affixed to each. Δt is set to be the smallest interval at which changes are made to the vehicle control inputs v and s, allowing the motion model of Section 3.4 to be used. After a step of movement, a new state estimate and covariance are produced:

$$\hat{\mathbf{x}}(k+1|k) = \begin{pmatrix} \mathbf{f}_v(\mathbf{x}_v(k|k), \mathbf{u}(k) \\ \hat{\mathbf{y}}_1(k|k) \\ \hat{\mathbf{y}}_2(k|k) \\ \vdots \end{pmatrix} \tag{5.10}$$

$$\mathtt{P}(k+1|k) = \begin{bmatrix} \frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v}\mathtt{P}_{xx}(k|k)\frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v}^\top + \mathtt{Q}(k) & \frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v}\mathtt{P}_{xy_1}(k|k) & \frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v}\mathtt{P}_{xy_2}(k|k) & \dots \\ \mathtt{P}_{y_1x}(k|k)\frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v}^\top & \mathtt{P}_{y_1y_1}(k|k) & \mathtt{P}_{y_1y_2}(k|k) & \dots \\ \mathtt{P}_{y_2x}(k|k)\frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v}^\top & \mathtt{P}_{y_2y_1}(k|k) & \mathtt{P}_{y_2y_2}(k|k) & \dots \\ \vdots & \vdots & \vdots & \end{bmatrix} \tag{5.11}$$

where $\mathbf{f}_v$ and $\mathtt{Q}(k)$ are as defined in Equation 3.22. This new covariance matrix is formulated from the usual EKF prediction rule $\mathtt{P}(k+1|k) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\mathtt{P}(k|k)\frac{\partial \mathbf{f}}{\partial \mathbf{x}}^\top + \mathtt{Q}(k)$, where $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ is the full state transition Jacobian:

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v} & \mathtt{0} & \mathtt{0} & \dots \\ \mathtt{0} & \mathtt{I} & \mathtt{0} & \dots \\ \mathtt{0} & \mathtt{0} & \mathtt{I} & \dots \\ \vdots & \vdots & \vdots & \end{bmatrix} .$$

5.3.4 Predicting a Measurement and Searching

Making a measurement of a feature i consists of determining its position relative to the robot: technically, the components in the robot frame $C0$ of the vector $\mathbf{h}_G$ from the head centre to the feature. The function of the true robot and feature positions giving the Cartesian components of this vector is:

$$\mathbf{h}_{Gi}^{C0} = \begin{pmatrix} h_{Gix}^{C0} \\ h_{Giy}^{C0} \\ h_{Giz}^{C0} \end{pmatrix} = \begin{pmatrix} \cos\phi(X_i - x) - \sin\phi(Z_i - z) \\ Y_i - H \\ \sin\phi(X_i - x) + \cos\phi(Z_i - z) \end{pmatrix} . \tag{5.12}$$

The predicted measurement of a feature at the current robot position is therefore calculated from the above equation with the state estimates $\hat{\mathbf{x}}_v$ and $\hat{\mathbf{y}}_i$ substituted for $\mathbf{x}_v$ and $\mathbf{y}_i$. When a measurement of this vector is made, the innovation covariance (which is the covariance of the difference between the true and measured values) will be given by:

$$\begin{aligned} \mathtt{S}_{h_G i} &= \frac{\partial \mathbf{h}_{Gi}}{\partial \mathbf{x}}\mathtt{P}\frac{\partial \mathbf{h}_{Gi}}{\partial \mathbf{x}}^\top + \mathtt{R}_L \\ &= \frac{\partial \mathbf{h}_{Gi}}{\partial \mathbf{x}_v}\mathtt{P}_{xx}\frac{\partial \mathbf{h}_{Gi}}{\partial \mathbf{x}_v}^\top + 2\frac{\partial \mathbf{h}_{Gi}}{\partial \mathbf{x}_v}\mathtt{P}_{xy_i}\frac{\partial \mathbf{h}_{Gi}}{\partial \mathbf{y}_i}^\top + \frac{\partial \mathbf{h}_{Gi}}{\partial \mathbf{y}_i}\mathtt{P}_{y_iy_i}\frac{\partial \mathbf{h}_{Gi}}{\partial \mathbf{y}_i}^\top + \mathtt{R}_L . \end{aligned} \tag{5.13}$$

$\mathtt{R}_L$ is the measurement noise (see Section 5.3.5) transformed into Cartesian measurement space.

To measure the feature, the active head is driven to the angles necessary for fixation on this estimated relative feature location (calculated as described in Section 3.3.3). In camera centred coordinates (reference frames L and R respectively), the vectors $\mathbf{h}_L$ and $\mathbf{h}_R$ from the camera optic centres to the feature and their covariances $\mathtt{P}_{h_L}$ and $\mathtt{P}_{h_R}$ (calculated by transforming $\mathtt{S}_{hGi}$) are formed. Both of these vectors will have zero x and y components since at fixation the z-axis-defining optic axes pass through the feature. Considering the left camera, image projection is defined by the the usual equations:

$$u_L = -fk_u\frac{h^L_{Lx}}{h^L_{Lz}} + u_0 \quad \text{and} \quad v_L = -fk_v\frac{h^L_{Ly}}{h^L_{Lz}} + v_0 \ . \tag{5.14}$$

The covariance matrix of the image vector $\mathbf{u}_L = \begin{pmatrix} u_L \\ v_L \end{pmatrix}$ is given by $\mathtt{U}_L = \frac{\partial \mathbf{u}_L}{\partial \mathbf{h}^L_L}\mathtt{P}_{h^L_L}\frac{\partial \mathbf{u}_L}{\partial \mathbf{h}^L_L}^\top$. The value of the Jacobian at $h^L_{Lx} = h^L_{Ly} = 0$ is

$$\frac{\partial \mathbf{u}_L}{\partial \mathbf{h}^L_L} = \begin{bmatrix} \frac{-fk_u}{h_{Lz}} & 0 & 0 \\ 0 & \frac{-fk_v}{h_{Lz}} & 0 \end{bmatrix} .$$

Specifying a number of standard deviations (typically 3), $\mathtt{U}_L$ defines an ellipse in the left image which can be searched for the feature patch representing the feature in question. The same procedure is followed in the right image (Figure 4.2 shows some typically generated search regions). Limiting search for feature patches to these small areas maximises computational efficiency and minimises the chance of obtaining mismatches. If the feature is successfully matched in both images, the final value of the measurement $\mathbf{h}^{C0}_{Gi}$ is calculated from the expression in Section 3.3.3.

5.3.5 Updating the State Vector After a Measurement

Before processing a measurement it is transformed into an angular form:

$$\mathbf{h}_i = \begin{pmatrix} \alpha_i \\ e_i \\ \gamma_i \end{pmatrix} = \begin{pmatrix} \tan^{-1}\frac{h_{Gix}}{h_{Giz}} \\ \tan^{-1}\frac{h_{Giy}}{h_{Gi\rho}} \\ \tan^{-1}\frac{I}{2h_{Gi}} \end{pmatrix} , \tag{5.15}$$

where h_{Gi} is the length of vector $\mathbf{h}_{Gi}$ and $h_{Gi\rho} = \sqrt{{h_{Gix}}^2 + {h_{Giz}}^2}$ is its projection onto the xz plane. I is the inter-ocular separation of the active head. These angles represent the pan, elevation and vergence angles respectively of an ideal active head positioned at the head centre and fixating the feature, "ideal" here meaning a head that does not have the offsets that the real head had — in terms of Figure 2.1(b), this would mean that vectors $\mathbf{n}$ and $\mathbf{c}$ would be zero, with $\mathbf{p}$ purely along the elevation axis. Now α_i, e_i and γ_i will be very close to the actual pan, elevation and vergence angles of the real active head at fixation, but accuracy is gained by taking account of all the head offsets in this way.

The reason for using an angular measurement representation at all is that it allows measurement noise to be represented as a constant, diagonal matrix. The largest error in measurements is in the accuracy with which features can be located in the image centre — the rule used is that a successful fixation lock-on has been achieved when the feature

is located within a radius of two pixels from the principal point in both images. This represents an angular uncertainty of around 0.3° (see Section 4.2.2), and Gaussian errors of this standard deviation are assigned to α_i, e_i and γ_i. The angular errors in measurements from the head axis encoders, which have repeatabilities of size 0.005°, are much smaller than this and can be neglected. The measurement noise covariance matrix is therefore:

$$\mathtt{R} = \begin{bmatrix} \Delta\alpha^2 & 0 & 0 \\ 0 & \Delta e^2 & 0 \\ 0 & 0 & \Delta\gamma^2 \end{bmatrix} . \tag{5.16}$$

With a diagonal $\mathtt{R}$, measurements α_i, e_i and γ_i are independent. This has two advantages: first, potential problems with bias are removed from the filter update by representing measurements in a form where the noise can closely be modelled as Gaussian. Second, the measurement vector $\mathbf{h}_i$ can be decoupled, and scalar measurement values used to update the filter in sequence. This is computationally beneficial since it is now not necessary to invert any matrices in the update calculation. For each scalar part of the measurement h_i (where h_i is one of α_i, e_i, γ_i for the current feature of interest), the Jacobian

$$\frac{\partial h_i}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial h_i}{\partial \mathbf{x}_v} & \mathtt{0}^\top & \dots & \mathtt{0}^\top & \frac{\partial h_i}{\partial \mathbf{y}_i} & \mathtt{0}^\top & \dots \end{pmatrix} .$$

is formed. This row matrix has non-zero elements only at locations corresponding to the state of the robot and the feature in question, since $h_i = h_i(\mathbf{x}_v, \mathbf{y}_i)$. The scalar innovation variance S is calculated as:

$$S = \frac{\partial h_i}{\partial \mathbf{x}} \mathtt{P} \frac{\partial h_i}{\partial \mathbf{x}}^\top + R = \frac{\partial h_i}{\partial \mathbf{x}_v} \mathtt{P}_{xx} \frac{\partial h_i}{\partial \mathbf{x}_v}^\top + 2\frac{\partial h_i}{\partial \mathbf{x}_v} \mathtt{P}_{xy_i} \frac{\partial h_i}{\partial \mathbf{y}_i}^\top + \frac{\partial h_i}{\partial \mathbf{y}_i} \mathtt{P}_{y_i y_i} \frac{\partial h_i}{\partial \mathbf{y}_i}^\top + R , \tag{5.17}$$

where $\mathtt{P}_{xx}$, $\mathtt{P}_{xy_i}$ and $\mathtt{P}_{y_i y_i}$ are 3×3 blocks of the current state covariance matrix $\mathtt{P}$, and R is the scalar measurement noise variance ($\Delta\alpha^2$, Δe^2 or $\Delta\gamma^2$) of the measurement. The Kalman gain $\mathtt{W}$ can then be calculated and the filter update performed in the usual way:

$$\mathtt{W} = \mathtt{P} \frac{\partial h_i}{\partial \mathbf{x}}^\top S^{-1} = S^{-1} \begin{pmatrix} \mathtt{P}_{xx} \\ \mathtt{P}_{y_1 x} \\ \mathtt{P}_{y_2 x} \\ \vdots \end{pmatrix} \frac{\partial h_i}{\partial \mathbf{x}_v}^\top + S^{-1} \begin{pmatrix} \mathtt{P}_{xy_i} \\ \mathtt{P}_{y_1 y_i} \\ \mathtt{P}_{y_2 y_i} \\ \vdots \end{pmatrix} \frac{\partial h_i}{\partial \mathbf{y}_i}^\top \tag{5.18}$$

$$\hat{\mathbf{x}}_{new} = \hat{\mathbf{x}}_{old} + \mathtt{W}(z_i - h_i) \tag{5.19}$$

$$\mathtt{P}_{new} = \mathtt{P}_{old} - \mathtt{W} S \mathtt{W}^\top . \tag{5.20}$$

z_i is the actual measurement of the quantity obtained from the head, and h_i is the prediction. Since S is scalar, S^{-1} is simply $\frac{1}{S}$. This update is carried out sequentially for each scalar element of the measurement.

5.3.6 Initialising a New Feature

When an unknown feature is observed for the first time, a vector measurement $\mathbf{h}_G$ is obtained of its position relative to the head centre, and its state is initialised to

$$\mathbf{y}_i = \begin{pmatrix} x + h_{Gix}\cos\phi + h_{Giz}\sin\phi \\ H + h_{Giy} \\ z - h_{Gix}\sin\phi + h_{Giz}\cos\phi \end{pmatrix} . \tag{5.21}$$

Jacobians $\frac{\partial \mathbf{y}_i}{\partial \mathbf{x}_v}$ and $\frac{\partial \mathbf{y}_i}{\partial \mathbf{h}_G}$ are calculated and used to update the total state vector and covariance (assuming for example's sake that two features are known and the new one becomes the third):

$$\mathbf{x}_{new} = \begin{pmatrix} \mathbf{x}_v \\ \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_i \end{pmatrix} \tag{5.22}$$

$$\mathtt{P}_{new} = \begin{bmatrix} \mathtt{P}_{xx} & \mathtt{P}_{xy_1} & \mathtt{P}_{xy_2} & \mathtt{P}_{xx}\frac{\partial \mathbf{y}_i}{\partial \mathbf{x}_v}^\top \\ \mathtt{P}_{y_1x} & \mathtt{P}_{y_1y_1} & \mathtt{P}_{y_1y_2} & \mathtt{P}_{y_1x}\frac{\partial \mathbf{y}_i}{\partial \mathbf{x}_v}^\top \\ \mathtt{P}_{y_2x} & \mathtt{P}_{y_2y_1} & \mathtt{P}_{y_2y_2} & \mathtt{P}_{y_2x}\frac{\partial \mathbf{y}_i}{\partial \mathbf{x}_v}^\top \\ \frac{\partial \mathbf{y}_i}{\partial \mathbf{x}_v}\mathtt{P}_{xx} & \frac{\partial \mathbf{y}_i}{\partial \mathbf{x}_v}\mathtt{P}_{xy_1} & \frac{\partial \mathbf{y}_i}{\partial \mathbf{x}_v}\mathtt{P}_{xy_2} & \frac{\partial \mathbf{y}_i}{\partial \mathbf{x}_v}\mathtt{P}_{xx}\frac{\partial \mathbf{y}_i}{\partial \mathbf{x}_v}^\top + \frac{\partial \mathbf{y}_i}{\partial \mathbf{h}_G}\mathtt{R}_L\frac{\partial \mathbf{y}_i}{\partial \mathbf{h}_G}^\top \end{bmatrix} \tag{5.23}$$

where $\mathtt{R}_L$ is the measurement noise $\mathtt{R}$ transformed into Cartesian measurement space.

It should be noted that a small amount of bias is introduced into the map in initialising features in this simple way, due to the non-linearity of the measurement process. The effect of this is small, however, once repeated measurements have been made of the feature using the unbiased angular measurement method of regular updates.

5.3.7 Deleting a Feature

A similar Jacobian calculation shows that deleting a feature from the state vector and covariance matrix is a simple case of removing the rows and columns which contain it. An example in a system where the second of three known features is deleted would be:

$$\begin{pmatrix} \mathbf{x}_v \\ \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \end{pmatrix} \rightarrow \begin{pmatrix} \mathbf{x}_v \\ \mathbf{y}_1 \\ \mathbf{y}_3 \end{pmatrix} \quad , \quad \begin{bmatrix} \mathtt{P}_{xx} & \mathtt{P}_{xy_1} & \mathtt{P}_{xy_2} & \mathtt{P}_{xy_3} \\ \mathtt{P}_{y_1x} & \mathtt{P}_{y_1y_1} & \mathtt{P}_{y_1y_2} & \mathtt{P}_{y_1y_3} \\ \mathtt{P}_{y_2x} & \mathtt{P}_{y_2y_1} & \mathtt{P}_{y_2y_2} & \mathtt{P}_{y_2y_3} \\ \mathtt{P}_{y_3x} & \mathtt{P}_{y_3y_1} & \mathtt{P}_{y_3y_2} & \mathtt{P}_{y_3y_3} \end{bmatrix} \rightarrow \begin{bmatrix} \mathtt{P}_{xx} & \mathtt{P}_{xy_1} & \mathtt{P}_{xy_3} \\ \mathtt{P}_{y_1x} & \mathtt{P}_{y_1y_1} & \mathtt{P}_{y_1y_3} \\ \mathtt{P}_{y_3x} & \mathtt{P}_{y_3y_1} & \mathtt{P}_{y_3y_3} \end{bmatrix} . \tag{5.24}$$

5.3.8 Zeroing the Coordinate Frame

It is possible at any time to re-zero the world coordinate frame at the current robot position. The new state becomes:

$$\mathbf{x}_{new} = \begin{pmatrix} \mathbf{x}_{vnew} \\ \mathbf{y}_{1new} \\ \mathbf{y}_{2new} \\ \vdots \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{h}_{G1} + \mathbf{H} \\ \mathbf{h}_{G2} + \mathbf{H} \\ \vdots \end{pmatrix} , \tag{5.25}$$

where $\mathbf{h}_{Gi}$ is the current vector from the head centre to feature i as given in Equation 5.12, and $\mathbf{H}$ is the constant vector describing the vertical offset from the ground plane to the

head centre. To calculate the new state covariance we form the sparse Jacobian matrix:

$$\frac{\partial \mathbf{x}_{new}}{\partial \mathbf{x}_{old}} = \begin{bmatrix} 0 & 0 & 0 & \dots \\ \frac{\partial \mathbf{h}_{G1}}{\partial \mathbf{x}_v} & \frac{\partial \mathbf{h}_{G1}}{\partial \mathbf{y}_1} & 0 & \dots \\ \frac{\partial \mathbf{h}_{G2}}{\partial \mathbf{x}_v} & 0 & \frac{\partial \mathbf{h}_{G2}}{\partial \mathbf{y}_2} & \dots \\ \vdots & \vdots & \vdots & \end{bmatrix}, \tag{5.26}$$

and calculate $\mathtt{P}_{new} = \frac{\partial \mathbf{x}_{new}}{\partial \mathbf{x}_{old}} \mathtt{P}_{old} \frac{\partial \mathbf{x}_{new}}{\partial \mathbf{x}_{old}}^{\top}$.

As discussed at the start of this chapter, this process does not provide any new information on the relative positions of the robot and features, but simply re-aligns the world coordinate frame so as to explicitly show the feature positions and covariances in this form. The information about the locations of the robot and features in the original world coordinate frame is lost.

5.4 Experiments

A series of experiments has been carried out to evaluate the localisation method described in the previous sections. As explained in Chapter 2, the software implementation of the algorithm is such that it may be used dually, either in simulation or in an actual robot implementation. The simulation mode has been invaluable during the development of the algorithm since it allows tests of particular parts of the system to be carried out under controlled conditions. However, it is not possible truly to simulate many of the conditions and unforseen problems arising when a robot attempts to navigate in the real world. It was therefore a necessity to carry out extensive experiments using the real implementation. In the following sections we describe how these were performed and the results obtained. Characterisation experiments A1 and A2 were followed by an extended experiment B with full ground-truth measurements, and an investigation into the advantages of carrying the full covariance matrix in C1 and C2.

5.4.1 Experimental Setup

The robot system was installed in a large laboratory which has a layout similar to potential areas of application in industry, with corridor-like pathways between benches, a high ceiling, and a variety of equipment as scenery, as shown in Figure 5.1(a). This area proved also to be suitable for the further experiments described in the following chapters where the robot made more autonomous motions than those described in this section.

Metric experiments into the performance of the localisation algorithm require that ground-truth measurements are made of the actual positions in the world of the robot and features used so that comparisons can be made with the filter output. A grid was therefore marked out on the floor of a section of the laboratory, as shown in Figure 5.1(b). The regularly spaced lines (at intervals of 20cm), covering an area of floor of size approximately 7 metres by 2 metres, provided nearby reference points from which to measure the robot's exact position anywhere within this region. Since, as described in Chapter 3, the robot's position in the world is described relative to the location of the point on the active head called the head centre, a vertical pointer attached to a weighted plumb line was suspended

(a) (b)

Figure 5.1: (a) The laboratory in the University of Oxford's Engineering Science Department where experiments took place. The natural lighting and variety of different objects make this a challenging but realistic environment for a vision system to deal with. (b) A grid of spacing 20cm was marked out in one corridor-like area of the laboratory floor to aid with ground-truth measurements. The grid was not used by the vision system.

from the robot body directly below this point, and the position of the pointer over the grid was used to define the robot's location.

To measure the orientation of the robot, a low power laser was mounted centrally at the front of the vehicle. The laser beam was aligned with the forward direction of the robot. As shown in Figure 5.2, to measure the current robot orientation the point where the laser beam crossed the farthest part of the grid was located. Using this point (z_L, x_L) along with the measured location of the head centre above the grid, the robot's orientation (defined relative to the z axis) could be calculated as:

$$\phi = \tan^{-1}\left(\frac{x_L - x}{z_L - z}\right) . \tag{5.27}$$

All grid measurements were made to an estimated accuracy of $\pm$1cm. Neither grid nor laser was ever used by the visual navigation system: they were simply aids to ground-truth measurements.

In many of the experiments, the ground-truth locations of the features detected and used by the robot were also measured. However, this was difficult for two reasons. First, the features lay in three-dimensional positions which were not necessarily close to grid lines, so measurements were physically difficult to make. Second, and more fundamentally, it was sometimes difficult to determine which part of an object the robot was using as a particular landmark feature. This is an example of the abstraction problem, discussed in Chapter 1: why should the robot's representation of what constitutes a particular interesting feature correspond with something that a human can easily identify? All that is important for the robot in choosing a feature is that it is a stationary 3D point which can be matched from different viewpoints. To produce ground-truth feature measurements, therefore, an estimate of the point on an object viewed as the landmark location by the robot was made by comparing matched views of it at fixation (as in Figure 4.3).

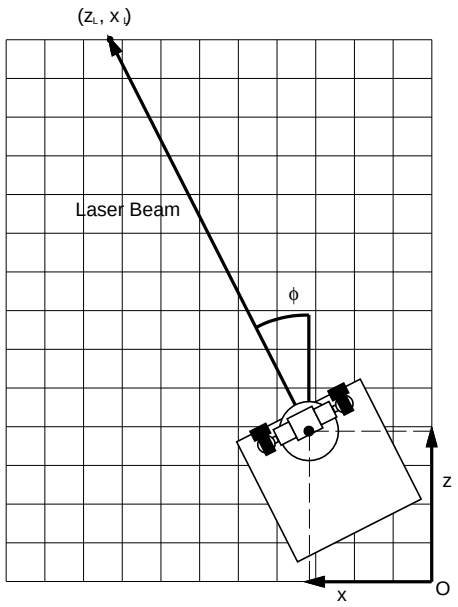


Figure 5.2: Measuring the orientation of the robot using a laser: the point (z_L, x_L) where the beam crosses the farthest line of the grid is measured along with (z, x), the location of the robot.

For these reasons, the "true" positions of features reported in the following sections must be treated as approximate.

5.4.2 Characterisation Experiments A1 and A2

In the first experiments carried out, the robot made motions in a step-by-step fashion in the region of a small set of features which it first measured and initialised into the filter from its starting point at the grid origin. The features were identified manually, by indicating regions in the robot's left image with the mouse pointer, rather than finding them automatically with the detector described in Section 4.1.1. This was so that landmarks in interesting and widely-spaced locations could be guaranteed.

At each step in its movement the robot stopped, made measurements of each of the features in turn, and updated estimates of its position and those of the features. The actual robot position and orientation relative to the grid were then manually measured and recorded.

For comparison, as well as calculating a state estimate based on the measurements made of all the features at each step, the robot produced estimates as if it had been measuring just each individual feature, and also as if no measurements had been made at all (this estimate being based solely on the vehicle odometry). The different trajectories produced in these ways could then all be compared with the ground-truth obtained from the manual grid measurements.

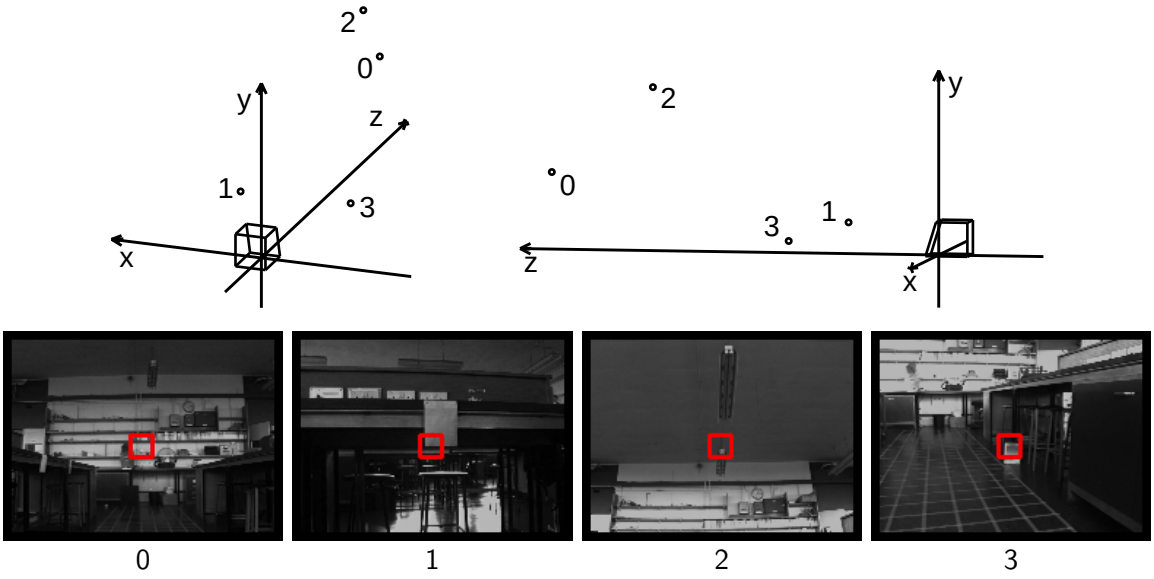


Figure 5.3: The four features used in the forward and back experiment A1, and 3D views of their true 3D locations. The robot is represented as a wedge shape in these two views, the first from just behind and to the right, and the second side-on from the left.

Experiment A1: Forward and Back

In this experiment the robot made a forward motion in a nominally straight line, and then reversed back towards its starting point. The movement was comprised of eighty steps altogether, each of which was around 5cm in size. Four landmarks were used, and are referred to by the labels 0–3. Their hand-measured (x, y, z) coordinates in 3D space are given below (in metre units):

0.	$(0.30, 1.49, 7.34)$	Markings on books on a shelf on the laboratory's back wall.
1.	$(0.98, 0.68, 1.55)$	A patch of paper attached to a cupboard to the left of the robot.
2.	$(-0.05, 3.10, 5.47)$	The end of a fluorescent lighting strip hanging from the ceiling.
3.	$(-0.70, 0.16, 2.99)$	A book on the floor to the right and ahead of the robot.

Figure 5.3 shows two different viewpoints of the 3D locations of the features relative to the robot's starting position, and fixated views of the patches saved.

Estimates produced using odometry only will be considered first: Figure 5.4 (which depicts the scene from an overhead viewpoint) shows the robot position estimated by the system as if no feature measurements were made at all during robot motion. Figure 5.4(a) is the starting situation, where the robot has made first observations of the four features and initialised them into the filter. The estimated feature locations are shown as grey points, and their uncertainties as grey ellipses. The ellipses have been calculated from the diagonal $\mathtt{P}_{y_i y_i}$ partitions of the total covariance matrix $\mathtt{P}$, the projection onto the 2D page of the ellipsoidal region within which it is expected that the true features lie at the 3σ level. These ellipses are very large, showing the large potential error in a single measurement, and are elongated in the direction of view, since this is where the uncertainty is greatest as discussed in Section 4.2.2.

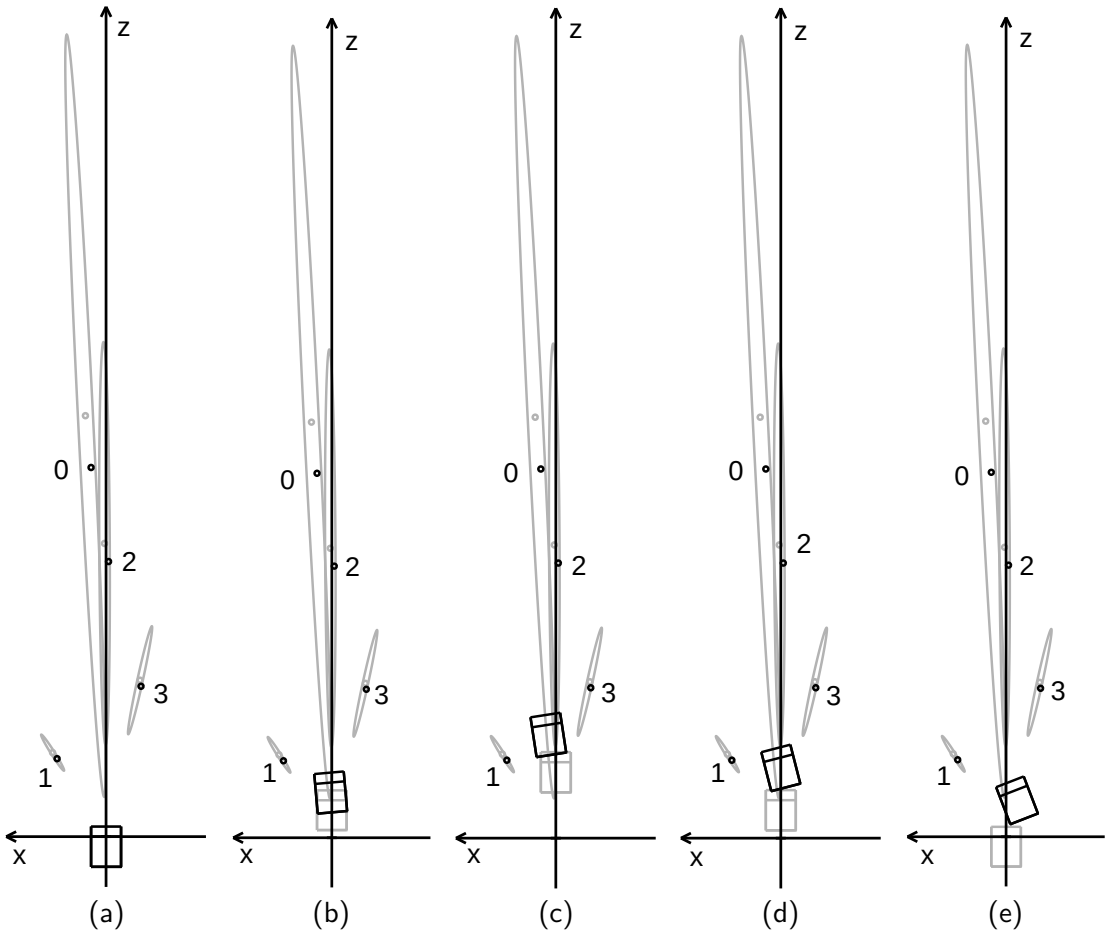


Figure 5.4: Estimating robot position using odometry only in experiment A1: grey shows system estimates of robot and feature locations, and black the actual measured values. From its starting position at (a) the robot moves forward through (b) to its most advanced location at (c), before reversing back through (d) to (e). It can be seen that the robot position estimate quickly diverges from the ground-truth due to poor odometry calibration.

The robot then moved forward through 20 steps to the position in (b) and a further 20 to (c). It can be seen that the estimated and true positions of the robot have diverged substantially: since it has been commanded to move directly forward at each step, the estimated position assumes simply that it has done this. However, actually, the robot has not been moving as commanded: it has travelled farther forward than expected, and turned slightly to the left. On the return journey, depicted through diagrams (d) and (e), further drift occurred: now the robot had turned more to the left, and not come as far back as commanded. The estimated location in (e) is that the robot has returned exactly to its origin, since the commands to it have been equal numbers of forward and backward steps of the same size.

In fact the robot is far from the origin: its actual position is:

$$\mathbf{x}_v = \begin{pmatrix} z \\ x \\ \phi \end{pmatrix} = \begin{pmatrix} 0.92 \\ -0.15 \\ 0.37 \end{pmatrix} .$$

The estimated position and covariance are:

$$\hat{\mathbf{x}}_v = \begin{pmatrix} 0.00 \\ 0.00 \\ 0.00 \end{pmatrix} \quad , \quad \mathtt{P}_{xx} = \begin{bmatrix} 0.0026 & 0.0000 & 0.0000 \\ 0.0000 & 0.0117 & -0.0116 \\ 0.0000 & -0.0116 & 0.0154 \end{bmatrix} .$$

Here, $\mathtt{P}_{xx}$ is the top-left 3×3 partition of the overall covariance matrix which represents the covariance of the estimates of the robot position parameters $\hat{z}$, $\hat{x}$ and $\hat{\phi}$. The other partitions of $\mathtt{P}$ are unchanged in this purely predictive update. The uncertainty represented by $\mathtt{P}_{xx}$ is large, as expected after a large motion with no measurements, but not large enough to account for the discrepancy of the estimate: the square root of the top-left element 0.0026, which is the standard deviation of $\hat{z}$, is 0.05, or 5cm, which is far less than the 92cm difference between z and $\hat{z}$.

The reason for the robot's true motion being so different from the commanded motion (which became the odometry-only estimate) in this experiment was that there was a lot of wheel-slip: in each forward step, the robot was moving at top speed through a distance of about 5cm, then rapidly stopping its driving wheel, but skidding through for another centimetre or two since the driving wheel is at the rear of the vehicle. In backward steps, slippage tended to occur as the robot accelerated rapidly away from a standstill, meaning that it travelled less far than expected. These effects led to a systematic error in odometry estimates, which is not accounted for in the vehicle motion model of Section 3.4. This problem was mitigated in later experiments by making the vehicle slow down more gradually. However, in this experiment it gave the filter a challenge to see how much the robot position estimate could be improved by tracking features even when the odometry was so misleading.

The estimated robot and feature positions calculated from the filter when making measurements of all four features at each step are shown in Figure 5.5. It can be seen that the estimated robot location is now close to the ground-truth throughout the motion — even though the odometry information is misleading, making repeated measurements of the positions of previously unknown features has enabled the robot to calculate its motion accurately.

This time the estimated position and covariance after the 80 steps are:

$$\hat{\mathbf{x}}_v = \begin{pmatrix} 0.86 \\ -0.17 \\ 0.37 \end{pmatrix} \quad , \quad \mathtt{P}_{xx} = \begin{bmatrix} 0.000150 & 0.000016 & 0.000005 \\ 0.000016 & 0.000009 & -0.000002 \\ 0.000005 & -0.000002 & 0.000025 \end{bmatrix} ,$$

compared with the ground-truth $z = 0.92, x = -0.15, \phi = 0.37$. Again, the top-left 3×3 partition of $\mathtt{P}$ represents uncertainty in the robot location, and again we see that due to the odometry problems it is smaller than it should be to account for the discrepancy, but this time the other elements of $\mathtt{P}$ have also changed as estimates of the location of the robot and that of each of the features become coupled by the repeated measurements. The ellipses describing the uncertainty in the estimates of the feature locations, produced from

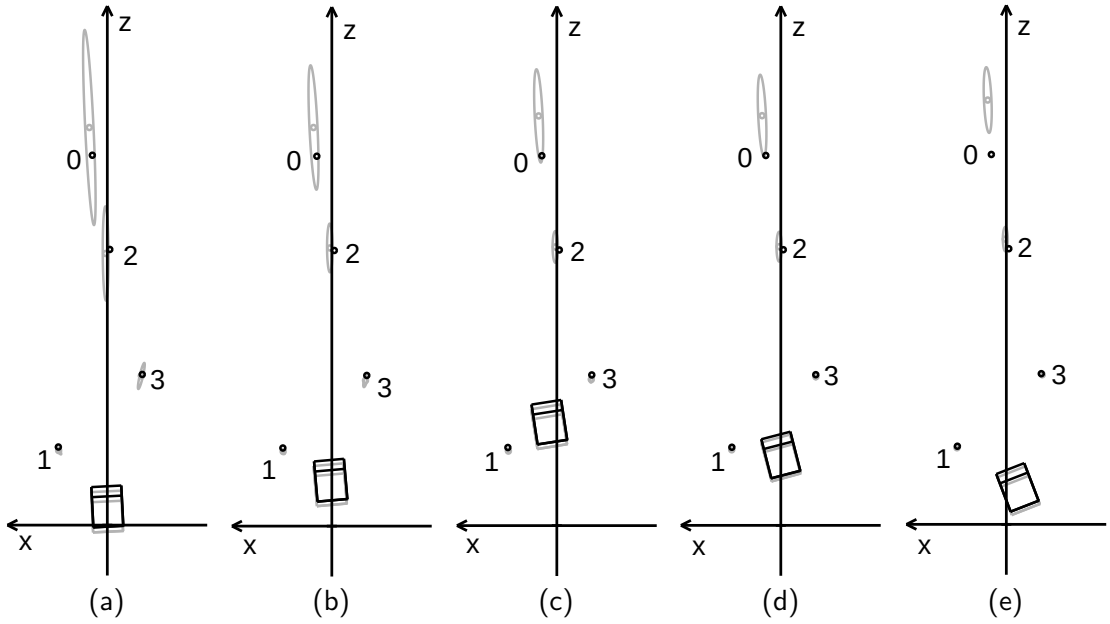


Figure 5.5: Estimating robot position when making measurements of all features in experiment A1: (a) is the state after 10 steps, and (b) – (e) show the situations after 20, 40, 60 and 80 steps respectively. The estimated robot position now follows the true location quite accurately, despite the very misleading odometry.

the diagonal $\mathtt{P}_{y_i y_i}$ partitions of $\mathtt{P}$, become smaller and smaller as more measurements are made. These ellipses should of course always enclose the true point positions, and this is not the case with feature 0 in Figure 5.5(e): the large odometry errors have led a biased estimate being made of this feature's position. It is of course always going to be difficult to measure the positions of features like this which are always distant from the robot, as is shown from the large uncertainty ellipses which remain for features 0 and 2 even after 80 measurements. However, with improved odometry in later experiments we shall see that estimation errors can be accounted for in an unbiased way.

Finally for this experiment we will look at the effect on the robot position estimation of making measurements of each of the features individually. Figure 5.6 shows the estimated states calculated as if the robot had made measurements of just one particular feature at each of its 80 steps. As expected, the uncertainty ellipse of that feature has shrunk in each case while the others have remained at their size from initialisation. It is interesting to compare the different estimated robot locations calculated: the information gained from measurements of a particular feature, and therefore the constraints placed on the robot's location depend on its position. In Figure 5.6(a), repeated measurements of feature 0, which is distant from the robot in the forward direction, have allowed the robot's orientation and x coordinate to be well estimated, but have not provided good information on its z coordinate — the measurement uncertainty is large in this direction, so the estimation process has placed more weight on the odometry data. Measurements of the other distant landmark, feature 2 as shown in (c), have similar results, but the accuracy in the z coordinate is better since this feature is closer to the robot and is also quite high up, so the change in elevation

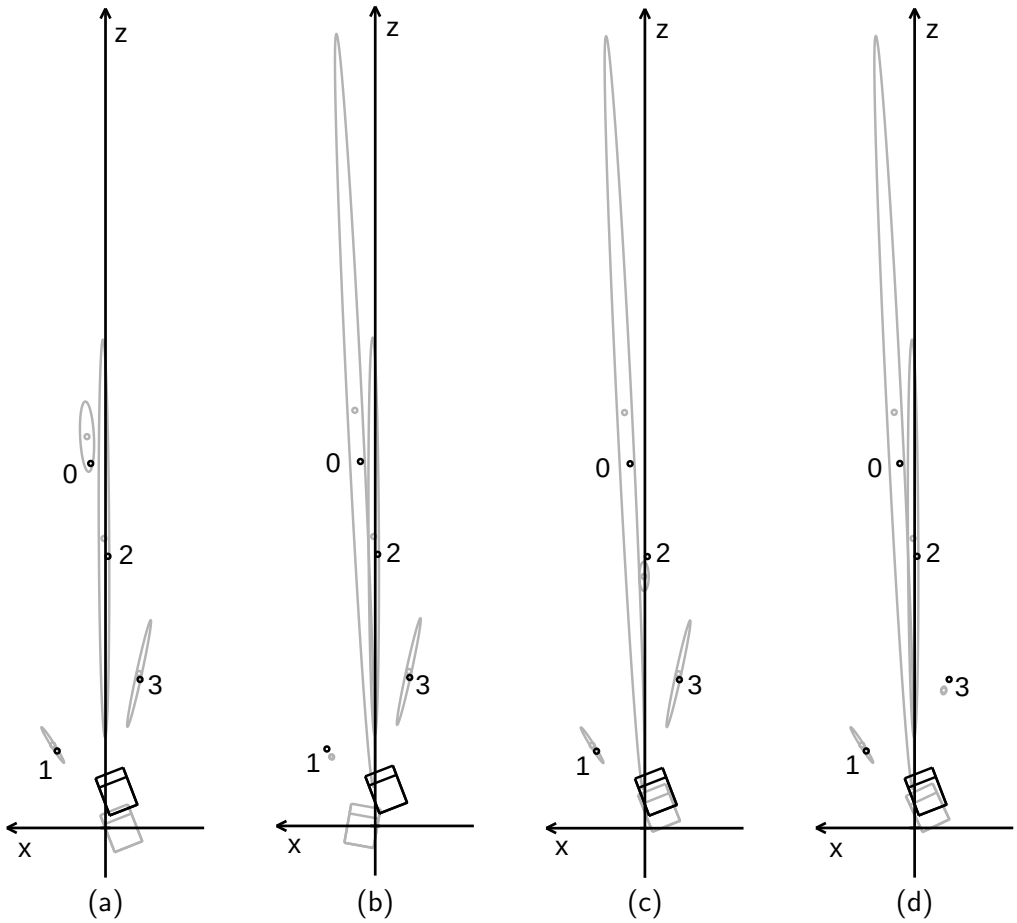


Figure 5.6: The estimated robot and feature locations in experiment A1 after 80 steps making exclusive measurements of features 0, 1, 2 and 3 respectively in (a), (b), (c) and (d).

angle needed to fixate it will provide depth information as well as the change in vergence.

Tracking feature 1 in (b), the closest feature to the robot, proves not to be very useful in this experiment. The information that measurements of this feature are able to provide accurately are the distance of the robot from it and the angle between the robot's forward direction and viewing angle to the feature. Referring back to Figure 5.4(e), it can been seen that final robot position calculated from the misleading odometry is in fact different from the true position in such a way as to lie along the ambiguous degree of freedom of these measurements: that is to say, a measurement made of feature 1 from either the true robot location or the estimated location would not differ much. The general nature of this ambiguity is made clearer in Figure 5.7. Measurements of feature 1 therefore do not help to resolve the estimation error. Measuring feature 3, as in Figure 5.6(d), is much more useful, since its position means that it can help with the particular direction of error in this experiment.

The main observation to be drawn from this comparison is that it is necessary to make measurements of multiple features to fully constrain estimates of the robot's motion. The

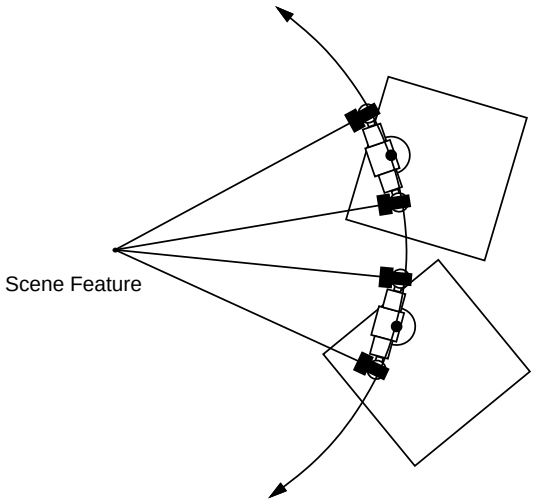


Figure 5.7: The different positions from which the robot will make the same measurements of the pan, elevation and vergence angles to fixate on a feature lie along a circle. The necessary robot orientation is fixed relative to a point on this locus by the pan angle, so there is only one degree of positional freedom. Measurements of the feature will not therefore reduce uncertainty along this direction.

estimates of Figure 5.5, where all the features have been measured, are much better than any of the estimates produced from the features individually. This is a point which will be returned to in Chapter 6 when the active choice of measurements is considered.

Experiment A2: Turning and Forward

In a second experiment, the robot made a turning, forward movement through a distance of around 2.5m while again making repeated measurements of four features. Figure 5.8 charts the progress of the estimation when measurements of all features were used, and compares this with the odometry-only estimate. It can be seen that the odometry information was now much more reliable (since wheel slippage had been reduced), and the robot position estimate in (e) is quite close to the ground-truth. It can be seen from (d), however, that the estimate obtained when making measurements is much better still.

5.4.3 Experiment B: An Extended Run

In typical navigation, the robot will not solely make use of landmarks detected from its starting position as in the experiments above: prominent landmarks will be initialised, measured and sometimes rejected from the map at all stages of the journey. Chapter 7 considers the automation of all these steps, but in this section an experiment is presented where the robot moved through a long distance, building up and using a realistic map, and stopping at frequent step intervals so that ground-truth measurements of its position could be made. The key aim of the experiment was to verify the localisation algorithm over a long run, and in particular check its ability to re-find features after long periods of neglect and "close the loop" on a round-trip journey.

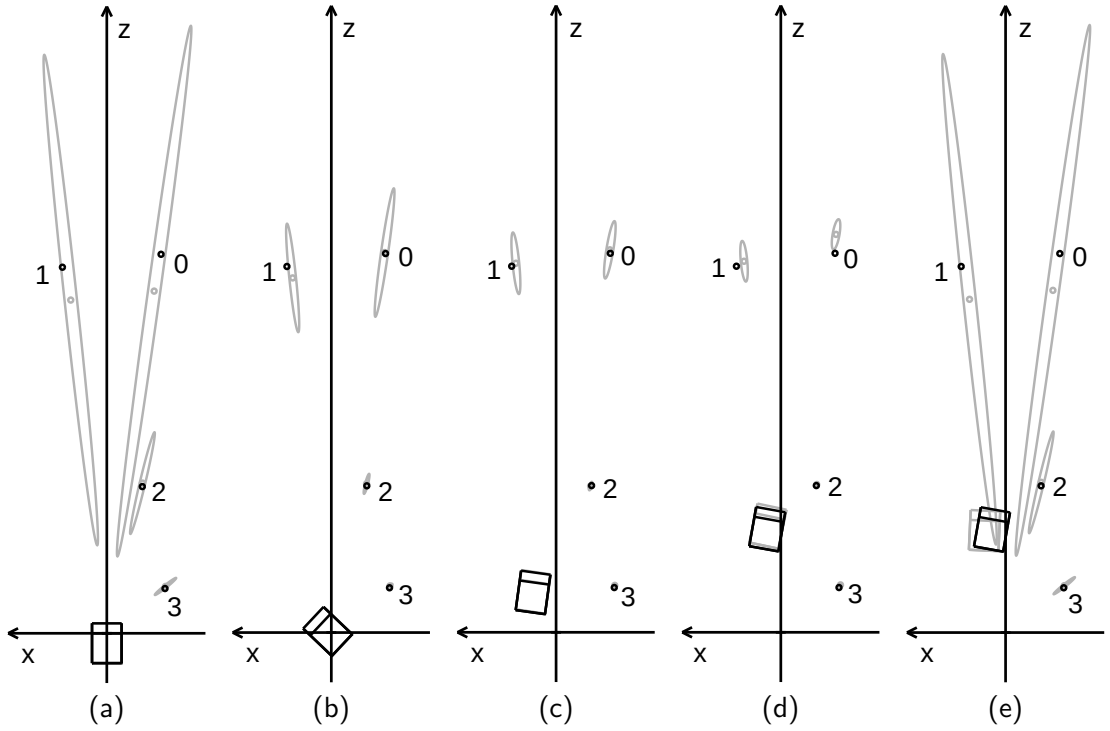


Figure 5.8: Estimating robot and feature locations during the turning, forward motion of experiment A2. (a) is the initial position when features have first been measured. (b), (c) and (d) show the state estimated from measurements of all features after 20, 45 and 79 movement steps respectively. In (e) the state estimated from odometry only after 79 steps is displayed.

From the robot's starting position, just two features were identified and initialised as landmarks. The landmark identification was carried out largely automatically, but some human input at this stage helped reliable features to be chosen. For each movement step (of approximate size 40cm), one feature was chosen, and 10 measurements were made of it during the motion (see Chapter 6 for an explanation of continuous tracking). After each step, the robot position was measured, and a new feature was chosen for the next step. New features were initialised when the existing ones went out of view.

15 features were used altogether as the robot moved twice up and down the gridded corridor-like area of the laboratory. The robot steered sharply at the ends of the corridor to turn through 180°. The farthest point reached was about 6m from the origin, so the total distance travelled was around 24m.

The features used are displayed and labelled in the order in which they were initialised in Figure 5.9. The progress of the robot can be followed in Figure 5.10, where an overhead view of the state at 12 numbered step counts is displayed, the convention of black for true and grey for estimated robot and feature locations again being used. Features are labelled in the diagrams shortly after they are first seen by the robot and initialised in the map, usually with large covariances. Some features, such as 3 and 5, were deleted automatically from the map because measurements had failed and they were "bad" in the senses described

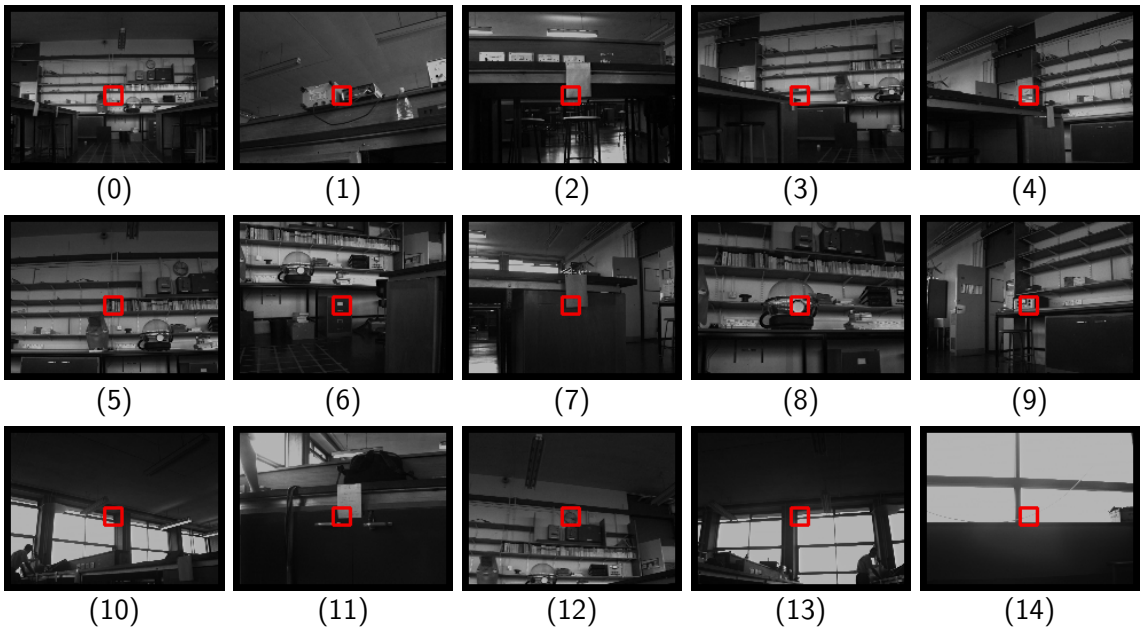


Figure 5.9: Fixated views of the 15 features used in the extended experiment B.

in Section 4.1. Feature 4 never shows up on the map because it was rejected very shortly after initialisation.

The robot moves up the corridor for the first time through steps (2), (7) and (10), before starting its first turn at step (14). It can be seen that the features initialised when the robot is at this end of the corridor, such as as numbers 5–9, have estimated positions which are much more uncertain than those initialised from near the origin such as 1 and 2 — their uncertainty ellipses are quickly reduced to a size which is too small to show up on the diagrams. This reflects the fundamental characteristic of map building discussed in Section 5.2.1: these features are far (in terms of the number of measurements implicitly compounded to estimate their locations) from the place where the coordinate frame is defined; the robot location itself is uncertain when they are measured, and so their locations are as well.

In step (22), the robot gets its first view of the back wall of the laboratory, and initialises feature 10 which is the corner of a window. Step (27) is key, because it provides an opportunity to re-measure an early feature: number 2 comes back into view. The result of this is that the estimate of the robot's position immediately improves — it had been drifting slightly up to step (22), but locks closely back on in (27). Also, the estimates of the locations of features initialised since measurements were last made of feature 2 improve, and their uncertainties reduce, due to their coupling to the robot state in the Kalman Filter — the uncertainty ellipses of features 6–9 and especially 10 visibly shrink. A further slight improvement occurs in step (32) when feature 1 is re-found.

Close to the origin, the robot turns in step (35) and heads back up the corridor. Far fewer new features need to be initialised now because the old ones are visible from this previously-followed route. Turning again at the far end of the corridor in (53) and back at the origin in (69), the robot halts at step (72), close to its starting position and orientation.

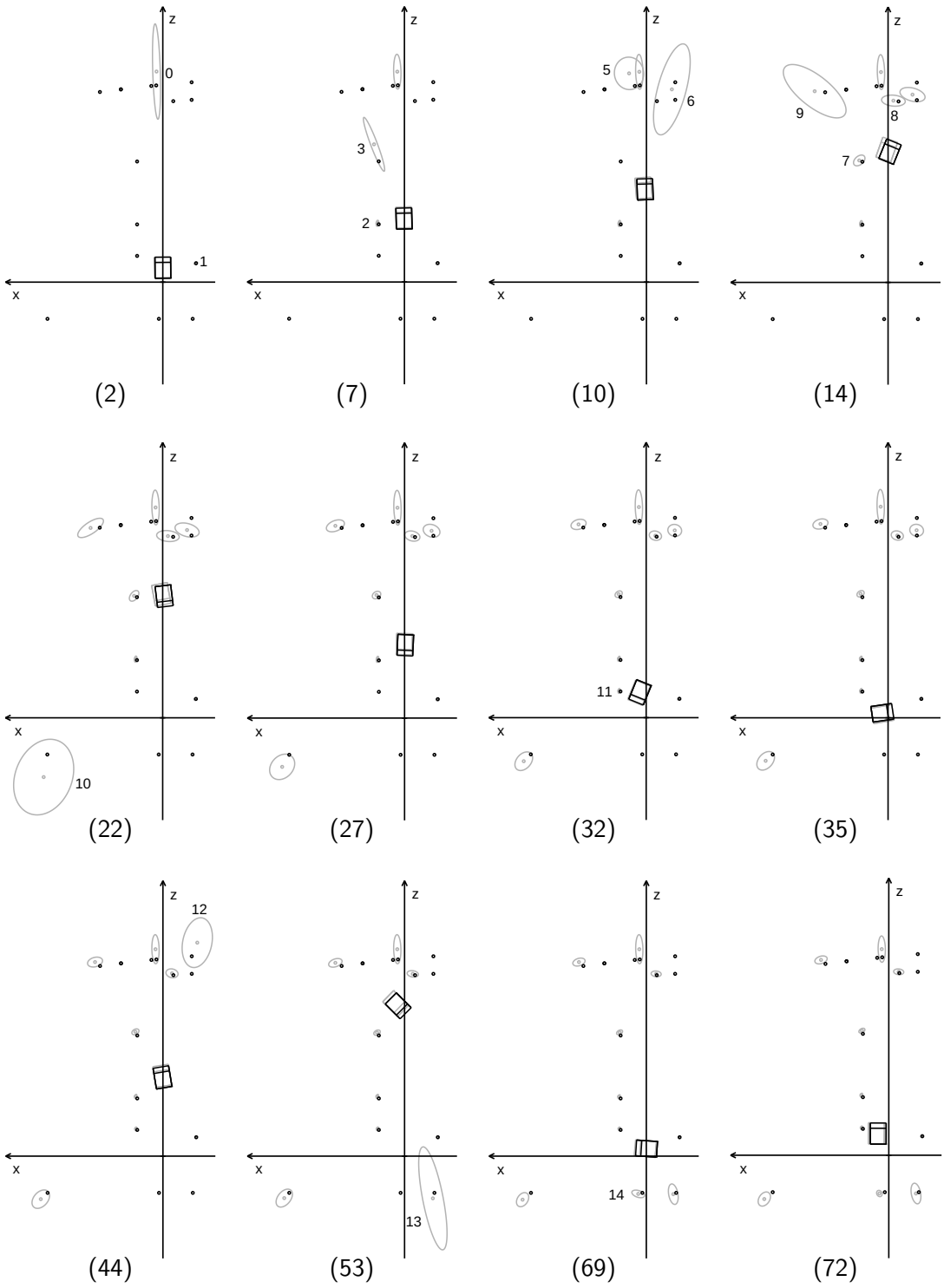


Figure 5.10: State snapshots (with step number in parenthesis) showing the robot travelling twice up and down the corridor-like area in the extended experiment B. The true robot position in black is tracked closely by the estimated position in grey throughout. Each numbered feature is labelled shortly after it is initialised.

The map of features generated sequentially through the experiment has now reached a fairly stable state, and there is no essential difference between landmarks detected early or late in the run.

To give a numerical idea of the localisation algorithm's performance, at step (14), where the robot was starting to turn for the first time at the far end of the corridor, the true and estimated robot locations (in metre and radian units) were:

$$\mathbf{x}_v = \begin{pmatrix} z \\ x \\ \phi \end{pmatrix} = \begin{pmatrix} 5.17 \\ -0.15 \\ -0.39 \end{pmatrix} \quad , \quad \hat{\mathbf{x}}_v = \begin{pmatrix} \hat{z} \\ \hat{x} \\ \hat{\phi} \end{pmatrix} = \begin{pmatrix} 5.25 \\ -0.02 \\ -0.34 \end{pmatrix} .$$

The covariance of the robot position estimate was:

$$\mathtt{P}_{xx} = \begin{bmatrix} 0.0035 & 0.0003 & 0.0002 \\ 0.0003 & 0.0069 & 0.0033 \\ 0.0002 & 0.0033 & 0.0017 \end{bmatrix} .$$

Feature 9, which had just been initialised here and had a large uncertainty, had true and estimated positions:

$$\mathbf{y}_9 = \begin{pmatrix} X_9 \\ Y_9 \\ Z_9 \end{pmatrix} = \begin{pmatrix} 2.40 \\ 1.10 \\ 7.25 \end{pmatrix} \quad , \quad \hat{\mathbf{y}}_9 = \begin{pmatrix} \hat{X}_9 \\ \hat{Y}_9 \\ \hat{Z}_9 \end{pmatrix} = \begin{pmatrix} 2.80 \\ 1.04 \\ 7.29 \end{pmatrix} ,$$

and covariance

$$\mathtt{P}_{y_9 y_9} = \begin{bmatrix} 0.1590 & 0.0186 & 0.0962 \\ 0.0186 & 0.0031 & 0.0161 \\ 0.0962 & 0.0161 & 0.1150 \end{bmatrix} .$$

In each case, the large covariance accounted easily for the discrepancy within one or two standard deviations: the robot's x coordinate, for example, was incorrectly estimated by 13cm, and the square root of the (2, 2) element of $\mathtt{P}_{xx}$ was $\sqrt{0.0069} = 0.08$: $\hat{x}$ had a standard deviation of 8cm.

We will also examine the later step (32), where the robot had returned to near the origin and re-measured early features. Here the true and estimated robot locations were:

$$\mathbf{x}_v = \begin{pmatrix} z \\ x \\ \phi \end{pmatrix} = \begin{pmatrix} 0.78 \\ 0.31 \\ 2.76 \end{pmatrix} \quad , \quad \hat{\mathbf{x}}_v = \begin{pmatrix} \hat{z} \\ \hat{x} \\ \hat{\phi} \end{pmatrix} = \begin{pmatrix} 0.77 \\ 0.35 \\ 2.75 \end{pmatrix} .$$

The covariance of the robot position estimate was:

$$\mathtt{P}_{xx} = \begin{bmatrix} 0.00096 & -0.00001 & -0.00052 \\ -0.00001 & 0.00011 & 0.00002 \\ -0.00052 & 0.00002 & 0.00035 \end{bmatrix} .$$

Feature 1, one of the features very close to the origin which was therefore known very accurately, had true and estimated positions:

$$\mathbf{y}_1 = \begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \end{pmatrix} = \begin{pmatrix} -1.27 \\ 1.29 \\ 0.72 \end{pmatrix} \quad , \quad \hat{\mathbf{y}}_1 = \begin{pmatrix} \hat{X}_1 \\ \hat{Y}_1 \\ \hat{Z}_1 \end{pmatrix} = \begin{pmatrix} -1.25 \\ 1.27 \\ 0.71 \end{pmatrix} ,$$

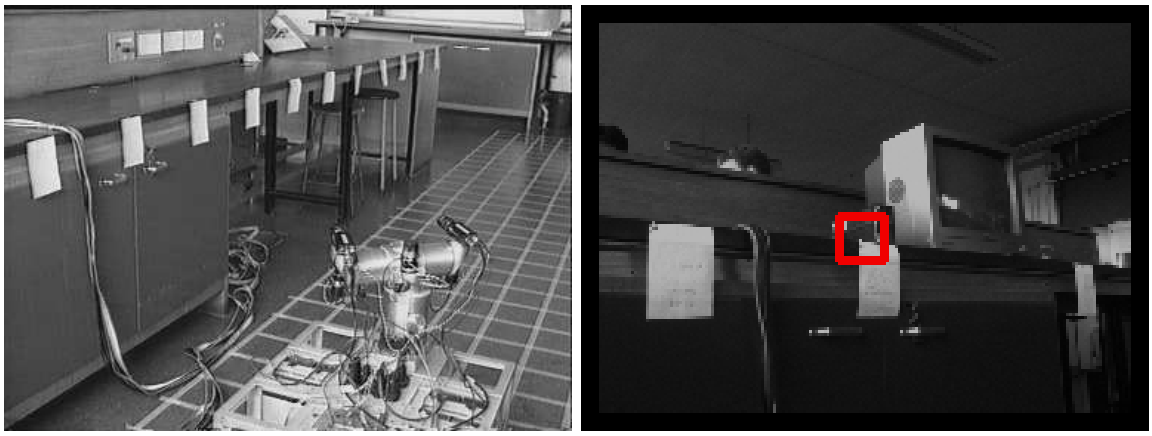


Figure 5.11: (a) Artificial paper beacons in a straight horizontal line (spaced by 40cm). (b) The top-left corner of each paper patch was used as a feature.

and covariance

$$\mathtt{P}_{y_1 y_1} = \begin{bmatrix} 0.00016 & -0.00007 & -0.00006 \\ -0.00007 & 0.00004 & 0.00004 \\ -0.00006 & 0.00004 & 0.00010 \end{bmatrix} .$$

It can be seen that the estimates were very accurate: to within a few centimetres for both the robot and feature positions. The covariances had shrunk to very small value, indeed slightly too small to account for the discrepancy between true and estimated values in a very small number of standard deviations (although in the case of feature 1, however, it must be remembered that "ground truth" measurements of feature locations are somewhat approximate). This indicates that the process or measurement noise in the filter may have been set too low, and the system had over-calculated its accuracy. However, system performance was not harmed, as can be seen from the algorithm's continued success as the robot moved through subsequent steps.

5.4.4 Experiments C1 and C2: The Advantages of Carrying the Full Covariance Matrix

An experiment was set up to explicitly demonstrate the advantages of the full covariance method of map-building over using uncoupled filters for the generation of maps of features which can be used for localisation over the equivalent of a "round trip" journey. As mentioned earlier, uncoupled filter approaches fail here due to their inability to re-find features seen early in the run and form a globally consistent map. The coupled approach means that automatic re-registration with the original coordinate frame occurs when early features can be measured again, as already demonstrated in the previous experiment.

An uncoupled-filters approach was easily implemented from our full-covariance system by zeroing off-diagonal partitions of the large matrix $\mathtt{P}$ after every prediction and measurement update. It can be shown that this is equivalent to direct implementation of the most sensible uncoupled approach: the robot state $\mathbf{x}_v$ and feature states $\mathbf{y}_i$ are now individual entities, with covariances described only by the diagonal parts $\mathtt{P}_{xx}$ and $\mathtt{P}_{y_i y_i}$ of the

full covariance matrix. In a prediction, only the robot state and covariance change. In a measurement update, only the state and covariance of the robot and the particular feature are affected; they become:

$$\hat{\mathbf{x}}_v(k+1|k+1) = \hat{\mathbf{x}}(k+1|k) + \mathtt{P}_{xx}\frac{\partial h_i}{\partial \mathbf{x}_v}^\top S^{-1}\nu \tag{5.28}$$

$$\mathtt{P}_{xx}(k+1|k+1) = \mathtt{P}_{xx}(k+1|k) - \mathtt{P}_{xx}(k+1|k)\frac{\partial h_i}{\partial \mathbf{x}_v}^\top S^{-1}\frac{\partial h_i}{\partial \mathbf{x}_v}\mathtt{P}_{xx}(k+1|k) \tag{5.29}$$

$$\hat{\mathbf{y}}_i(k+1|k+1) = \hat{\mathbf{y}}_i(k+1|k) + \mathtt{P}_{y_iy_i}\frac{\partial h_i}{\partial \mathbf{y}_i}^\top S^{-1}\nu \tag{5.30}$$

$$\mathtt{P}_{y_iy_i}(k+1|k+1) = \mathtt{P}_{y_iy_i}(k+1|k) - \mathtt{P}_{y_iy_i}(k+1|k)\frac{\partial h_i}{\partial \mathbf{y}_i}^\top S^{-1}\frac{\partial h_i}{\partial \mathbf{y}_i}\mathtt{P}_{y_iy_i}(k+1|k) \tag{5.31}$$

S, the measurement variance (*measurements still being split into their scalar components as in Section 5.3.5*), has a simpler form than in Equation 5.17:

$$S = \frac{\partial h_i}{\partial \mathbf{x}_v}\mathtt{P}_{xx}\frac{\partial h_i}{\partial \mathbf{x}_v}^\top + \frac{\partial h_i}{\partial \mathbf{y}_i}\mathtt{P}_{y_iy_i}\frac{\partial h_i}{\partial \mathbf{y}_i}^\top + R\ . \tag{5.32}$$

The effect of this S is that when performing updates of the robot state, the effective measurement uncertainty is the actual noise in the measurement plus the current uncertainty in the feature's position. When updating the estimate of the feature position, the effective measurement uncertainty is the noise of the measurement plus the uncertainty in the robot position. This is indeed the case, but what is not taken account of is possible coupling between uncertainty in the robot and feature estimates: for example, the position of neither may be known very well in the world coordinate frame, but the relative position is well determined. There is no way to represent this situation with uncoupled filters.

For the experiment, artificial beacons were created by affixing a row of paper markers to the bench at the side of the corridor (see Figure 5.11(a)). These beacons ensured that there was a regular supply of reliable features to measure along the forward and backward robot journeys. Also, the ground-truth locations of these landmarks could be measured accurately, and this made the coupled filter part of the experiment useful in itself because the accuracy of the system's estimation of point positions could be investigated reliably.

Starting from the grid origin, the robot was driven forward in a nominally straight line. Every second paper feature was initialised and tracked for a short while on this outward journey, which was comprised of steps of around 40cm as in the previous experiment. When the features were initialised by the robot, the selection of the original patch was done manually to ensure that an exact part of each paper patch (in this case the top-left corner as shown in Figure 5.11(b)) was agreed on by robot and human ground-truth measurer. The robot then reversed back down the corridor, tracking the features it had *not* previously seen. The aim was that it should return to its origin while tracking only recently acquired features, as would be the case in a looped movement around a rectangular layout of corridors for example (such a layout was not available in our laboratory). Once the robot had returned to near its starting position, it drove forward again, now attempting to re-measure features found early on, completing the loop on the motion and establishing that a reliable map had been formed. The whole experiment was carried out twice: once with the full coupled covariance filter, and once with the uncoupled implementation.

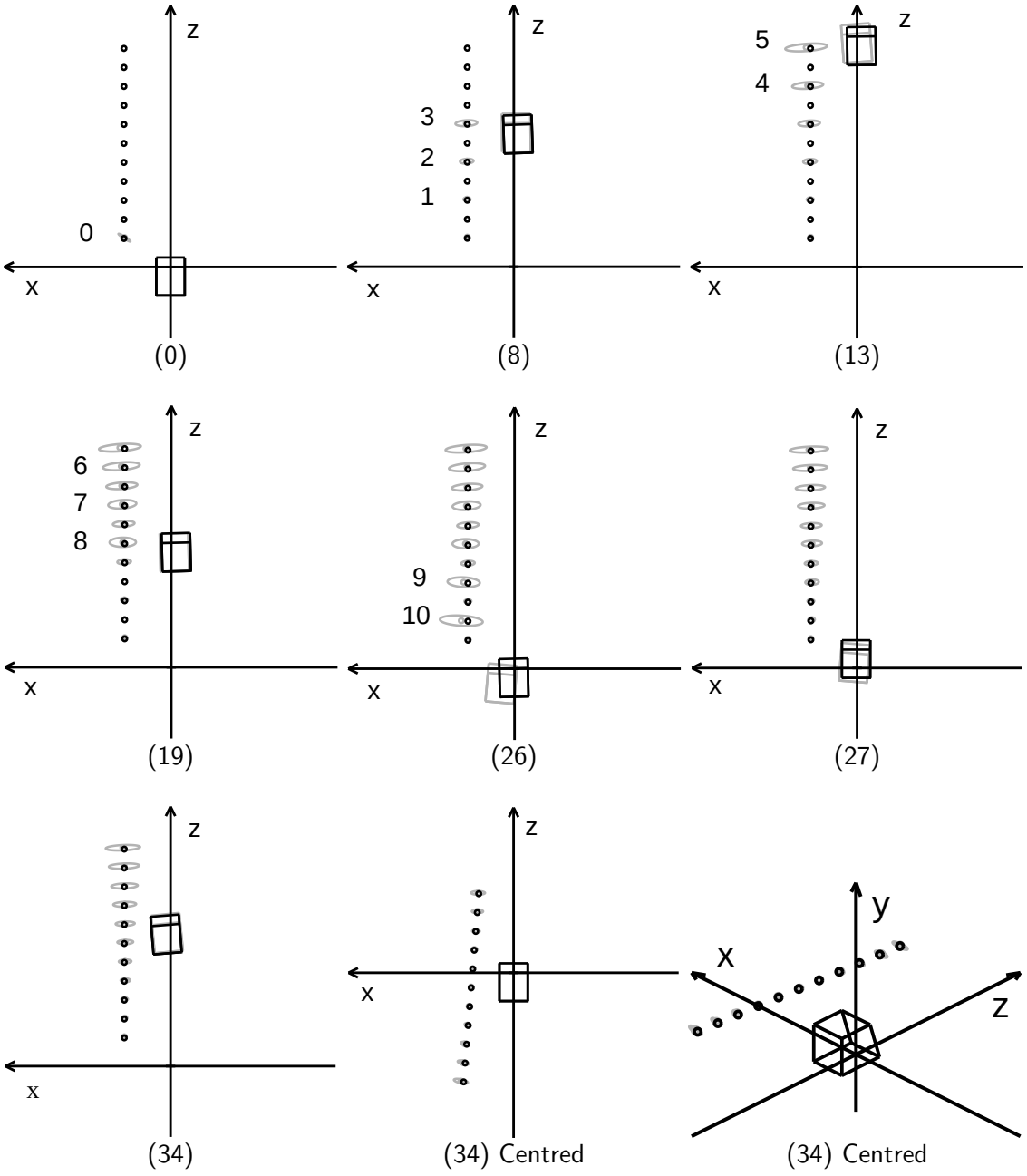


Figure 5.12: Steps from experiment C1 using fully coupled robot and feature estimates.

Experiment C1: Carrying the Full Covariance Matrix

Results from the coupled run are displayed in Figure 5.12, labelled with step and feature numbers. In the outward journey up to step (13), the robot position estimate starts to drift, its covariance grows, and the uncertainties in the positions of newly initialised features are large. This process continues as the robot returns to near the origin through steps (19) and (26) since only new features are tracked along the way. By now, the estimated robot

location is quite offset from the ground-truth. Step (27) is the first after an initial feature is re-seen: measurements are made of feature 0. The robot estimate immediately locks back well onto the true position. The estimate then remains good as the robot moves away from the origin again in step (34), now making measurements of other features.

The sign that the map-building has taken place consistently and without bias between features detected early and late in the run is the pattern of the feature uncertainty ellipses seen in steps (27) and (34): there is no distinction between the alternate ones measured in the forward and backward runs. This is an impressive feat from the algorithm: having references to well known features at both ends of the motion has allowed all the estimates made about features in between to be improved. What the pattern of uncertainty remaining, where uncertainty in the x direction grows in a regular way with distance from the origin, shows is the fundamental uncertainty in map-building from compounded measurements: because the features most distant from the origin can only be measured when the robot is at that end of the corridor (due to the limitations on range and angle in patch matching described in Section 4.1.2), estimates of their positions will be made uncertain due to the implicit combination of a large number of successive measurements which goes into estimating the robot's position in this area.

It must be remembered though that the uncertainty in the farther features is only with respect to the somewhat arbitrary global frame of reference defined at the robot's starting position: features 5 and 6 for instance, both have uncertainties which are large in the world frame, but their position relative to one another is well known, this information being held in the covariance partition $\mathtt{P}_{y_5 y_6}$. At step (34),

$$\mathtt{P}_{y_5 y_6} = \begin{bmatrix} 0.01290 & -0.00001 & -0.00057 \\ 0.00000 & 0.00000 & 0.00000 \\ -0.00052 & 0.00001 & 0.00034 \end{bmatrix} .$$

The large value 0.01290 for the top-left element of this matrix partition shows that the X coordinates of these features are well correlated.

The correlation between features is demonstrated clearly in the last two pictures of Figure 5.12 (the second a view from an angle as a reminder of the 3D nature of the map), where the state in step (34) is shown after the coordinate frame has been zeroed to the robot's current position as in Section 5.3.8. It can be seen that those features currently close to the robot, such as 3 and 8, are very well known in the new frame — their positions relative to the robot are certain because measurements need not be compounded to estimate this. The features with the largest uncertainties in the new frame are those distant from the robot in *both* directions — there is nothing special about the original coordinate frame.

Experiment C2: Using Uncoupled Covariances

The experiment was re-run, now using uncoupled filters for the vehicle and feature states as described above, but with similar robot actions. The state at various steps is shown in Figure 5.13. The state estimates seen in steps (7), (13) and (26) are similar in accuracy to those of experiment C1 — that is to say that the discrepancy between estimated robot and feature positions and ground truth is of around the same size. However, what has not happened correctly is growth in uncertainty in the robot state and that of the features initialised late in the journey: the location of feature 10 at step (26), for instance, is clearly

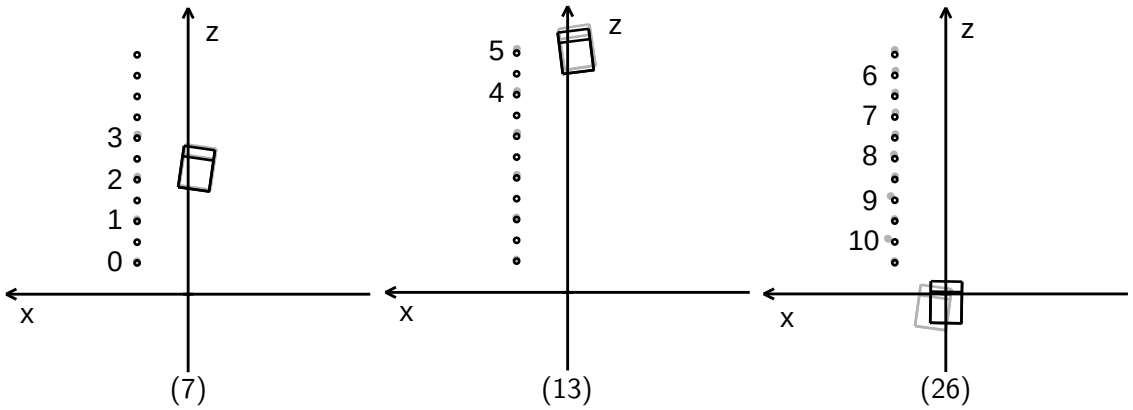


Figure 5.13: Steps from experiment C2 with uncoupled robot and feature estimates.

estimated quite incorrectly, but its uncertainty ellipse does not reflect this in the same way as that of feature 10 in experiment C1 (Figure 5.12, step (26)).

Only 3 state pictures are shown in Figure 5.13, since the result of this misleading filtering was that after step (26), the robot was unable to re-find the features found early-on, such as feature 0. The uncoupled filter had led the robot to overconfidently estimate its location when in fact there should have been a large uncertainty, and the feature did not lie in the image search ellipses generated. The robot was therefore never able to re-register itself with the original world frame as in experiment C1.

5.5 Conclusion

We have described an accurate and reliable approach to simultaneous map-building and localisation using active vision. The algorithm has been proven in experiments, and in particular its ability to recognise features after periods of neglect to re-register with an original coordinate frame is satisfying and essential to long-term navigation in confined regions.

6

Continuous Feature Tracking, and Developing a Strategy for Fixation

Active vision provides the capability to view and make measurements of features lying over a wide range of viewpoints. The previous chapter has shown how it can be used in a localisation and map-building system using landmark features: in experiments, accurate information on a new absolute robot position could be obtained from measurements of one or more of these features. However, we have not yet considered the strategy that should be followed by the active cameras: we know how to extract all the information from a particular measurement, but measurements of some features will be more valuable than those of others, or perhaps more convenient to obtain.

In particular, the active head allows the robot to *track* at fixation one feature in almost any viewing direction whenever the robot is moving, making repeated measurements of that feature as it does so which provide continuous navigation information. How this tracking is achieved will be considered in Section 6.1.

Maintaining a large map of features with full covariance storage, as described in the previous chapter, presents computational difficulties to fast tracking of features with straightforward full state updates at every step. In Section 6.2, a method is presented by which continuous tracking is never compromised by the computational burden of maintaining a large map, by updating in real-time only those parts of the state representation which are essential to the current tracking task. The effort to be expended at each step can be reduced to a constant amount. A small, constant amount of additional stored information allows the full state to be updated when the robot stops moving and has processing time to spare.

Section 6.3 considers the strategies that should be adopted to maximise the capabilities of the active head to make measurements of scene features and provide navigation information. Criteria based on the information content of measurements are developed to tackle

the questions of selecting between known features, selecting new features, and selecting between features while the robot is moving. These will form an essential part of the automatic map-building and localisation system to be presented in Chapter 7.

6.1 Continuous Tracking of Features

Tracking a feature consists of making repeated fixated measurements of it as the robot moves past. This task is of course similar to that of tracking a moving target from a stationary robot viewpoint, a problem which has been studied in depth already in active vision as described in Chapter 1. The situation is somewhat simplified, however, when the relative movement between robot and target is provided by the robot: the robot is under its own control, so its motion is at least approximately known. Tracking a freely manoeuvring target is difficult because although a model of its possible motions may be known, the "control inputs" are not. This is not the case when a robot tracks a stationary target.

Also, in the robot vehicle motion model of Section 3.4, it was shown that movements could be described directly in terms of the two control inputs velocity and steering angle — knowledge of these allows an estimate of a new position to be calculated from the previous position estimate without extra parameters. This might not be the case in a robot with more complex dynamics: suppose it was like a normal road car, where the control inputs are the steering angle and the position of the accelerator pedal, which is more closely related to the acceleration of the vehicle than its actual speed (as discussed by Maybank *et al.* [59] with regard to visual tracking of externally viewed cars) — in order to produce a new position estimate, it is necessary to know the current velocity as well as the current position. The velocity must therefore be part of the state vector of estimated quantities. Other vehicles may require higher-order derivatives to be estimated as well.

For these reasons, our tracking algorithm can proceed quasi-statically. This means that continuous tracking can be performed with the same filtering approach as would be used if the robot was moving in a step-by-step fashion, stopping each time a measurement was to be made. Assuming a certain inter-measurement interval, as the robot starts a movement, the prediction step of the filter produces an estimate of its new position at the next measurement point, and the expected relative position of the target feature is calculated. The active head is then driven to fixate on where the feature should be found, and the robot waits to make its measurement as the interval checkpoint on its trajectory is passed. As mentioned in Section 2.7, the vehicle odometry is used as the trigger for measurements: the trajectory is divided into steps of equal numbers of wheel encoder counts. This is better than using strict time intervals, since the distance travelled is what really matters, and possible fluctuations in velocity (particularly when the robot is just starting or stopping) will be unimportant. Because of this, the measurement frequency is not strictly constant, but in current implementation tracking is performed at approximately 5Hz (the odometry measurement intervals being dynamically set proportional to the robot velocity to keep this constant).

The actions taken in a step of the tracking loop, during which the robot drives continuously, are therefore:

1. Based on the current control inputs, perform the prediction step of the filter and estimate the new robot position.

Figure 6.1: Image sequences obtained from continuous fixation tracking of features.

2. Calculate the estimated head angles for fixation on the feature, and move the head to this position in readiness.
3. Wait until the robot has reached the correct "new" position according to its odometry.
4. Obtain new images from the cameras.
5. Perform correlation searching for the feature as in Section 4.1.2.
6. If the feature is found in both images, perform a measurement update of the filter.

Figure 6.1 shows image sequences obtained from one of the robot's cameras in periods of fixation tracking.

6.2 Updating Motion and Structure Estimates in Real Time

As explained in Chapter 5, when estimating multiple quantities in a Kalman Filter framework (in our case the positions of the robot and the features it uses as landmarks) , the best approach is to combine the estimates into a single state vector and to store the uncertainty in the estimates in an overall covariance matrix. However, a problem with proceeding in this way is that once the number of quantities being estimated becomes large (in our case the number of landmarks), the sizes of the state vector and in particular the covariance matrix make them unwieldy and computationally expensive to update at each step of the filter's evolution. This will clearly present a problem to a system with limited processing resources attempting to run at a high update rate, as with our robot when continuously tracking a feature while moving.

In this section we present an approach which under certain common circumstances will reduce the computation to be performed per time-step to a constant small amount so that real-time operation can be achieved *regardless of the size of the state vector*. The key to the method is to update only the parts of the state vector and covariance matrix which are necessary to the current real-time operation. Information about the generic way to update the other parts of the vector and matrix is stored at the same time (requiring a

small and constant amount of computation and storage) so that these parts can be fully updated at the end of the real-time run. The method is not an approximation, but a simple re-arrangement of the book-keeping of updating the estimates. The method will be presented in a formulation as general as possible to demonstrate its potential application to other map-building and localisation systems. Some of the notation and derivations from Chapter 5 will be re-stated in this general form for clarity. A summary of the details needed for implementation is given in Section 6.2.5.

6.2.1 Situation and Notation

We consider the general situation of estimating the locations of a moving sensor and several stationary features of which it is able to make measurements. The state of the system can be represented by the stacked vector

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_v \\ \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \end{pmatrix} , \tag{6.1}$$

where $\mathbf{x}_v$ is the location (or more generally the state) of the sensor, and $\mathbf{y}_i$ is the location of the ith feature.

Our current estimate of this true state is the vector $\hat{\mathbf{x}}$ with covariance matrix $\mathtt{P}$. These can be written in the partitioned forms:

$$\hat{\mathbf{x}} = \begin{pmatrix} \hat{\mathbf{x}}_v \\ \hat{\mathbf{y}}_1 \\ \hat{\mathbf{y}}_2 \\ \vdots \end{pmatrix} \quad , \quad \mathtt{P} = \begin{bmatrix} \mathtt{P}_{xx} & \mathtt{P}_{xy_1} & \mathtt{P}_{xy_2} & \dots \\ \mathtt{P}_{y_1x} & \mathtt{P}_{y_1y_1} & \mathtt{P}_{y_1y_2} & \dots \\ \mathtt{P}_{y_2x} & \mathtt{P}_{y_2y_1} & \mathtt{P}_{y_2y_2} & \dots \\ \vdots & \vdots & \vdots & \end{bmatrix} . \tag{6.2}$$

The partitioned sections of the covariance matrix represent the coupling of the uncertainties in the different parts of the estimated state. $\mathtt{P}_{xy_i}$ is the covariance matrix between the estimated robot state $\hat{\mathbf{x}}_v$ and that of one of the features $\hat{\mathbf{y}}_i$, while $\mathtt{P}_{y_iy_j}$ is that between the estimated locations of two features $\hat{\mathbf{y}}_i$ and $\hat{\mathbf{y}}_j$.

Recalling the standard Extended Kalman Filter prediction and update formulae from Section 5.1, we impose the simplifying specifics of the situation under consideration. These are:

1. The state transition function $\mathbf{f}$ has the form

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} \mathbf{f}_v(\mathbf{x}_v, \mathbf{u}) \\ \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \end{pmatrix} . \tag{6.3}$$

   This means simply that only the sensor moves in the scene, and not the features. Function $\mathbf{f}_v$ models the sensor movement — $\mathbf{u}$ is the control vector which may or may not be known depending on the exact scenario. The $\mathbf{y}_i$ feature parts of the state vector do not change in the state transition.

2. When a measurement is made of a feature, the measurement depends only on the current states of the sensor and that particular feature:

$$\mathbf{h}(\mathbf{x}) = \mathbf{h}_i(\mathbf{x}_v, \mathbf{y}_i) \ . \tag{6.4}$$

6.2.2 Prediction and Update

We will form the Kalman prediction and update equations for these circumstances, and then show what manipulations can be performed. Referring for form to some of the expressions derived in Section 5.3, we can first write down the following expressions for the prediction of the state after a motion and the covariance of that prediction:

$$\hat{\mathbf{x}}_{new} = \begin{pmatrix} \mathbf{f}_v(\hat{\mathbf{x}}_v, \mathbf{u}) \\ \hat{\mathbf{y}}_1 \\ \hat{\mathbf{y}}_2 \\ \vdots \end{pmatrix} , \tag{6.5}$$

$$\mathtt{P}_{new} = \begin{bmatrix} \frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v}\mathtt{P}_{xx}\frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v}^\top + \mathtt{Q} & \frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v}\mathtt{P}_{xy_1} & \frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v}\mathtt{P}_{xy_2} & \dots \\ \mathtt{P}_{y_1x}\frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v}^\top & \mathtt{P}_{y_1y_1} & \mathtt{P}_{y_1y_2} & \dots \\ \mathtt{P}_{y_2x}\frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v}^\top & \mathtt{P}_{y_2y_1} & \mathtt{P}_{y_2y_2} & \dots \\ \vdots & \vdots & \vdots & \end{bmatrix} . \tag{6.6}$$

$\mathtt{Q}$ is the covariance matrix of the process noise associated with the motion. Note that here we have dropped the usual "k" notation denoting the time-step referred to in Kalman Filter equations. For simplicity, we will just describe the ways the estimated state vector and its covariance change in predictions or updates.

We now consider updating the filter after a measurement $\mathbf{z}_i$ has been made of feature i. Since $\mathbf{h}_i$ is a function of $\mathbf{x}_v$ and $\mathbf{y}_i$ only, we can write:

$$\begin{aligned} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}} &= \begin{pmatrix} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v} & \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_1} & \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_2} & \dots & \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i} & \dots \end{pmatrix} \\ &= \begin{pmatrix} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v} & \mathtt{0} & \mathtt{0} & \dots & \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i} & \dots \end{pmatrix} . \end{aligned} \tag{6.7}$$

So:

$$\begin{aligned} \mathtt{P}\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}}^\top &= \begin{bmatrix} \mathtt{P}_{xx} & \mathtt{P}_{xy_1} & \mathtt{P}_{xy_2} & \dots \\ \mathtt{P}_{y_1x} & \mathtt{P}_{y_1y_1} & \mathtt{P}_{y_1y_2} & \dots \\ \mathtt{P}_{y_2x} & \mathtt{P}_{y_2y_1} & \mathtt{P}_{y_2y_2} & \dots \\ \vdots & \vdots & \vdots & \end{bmatrix} \begin{pmatrix} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top \\ \mathtt{0} \\ \mathtt{0} \\ \vdots \\ \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top \\ \vdots \end{pmatrix} \\ &= \begin{pmatrix} \mathtt{P}_{xx} \\ \mathtt{P}_{y_1x} \\ \mathtt{P}_{y_2x} \\ \vdots \end{pmatrix} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top + \begin{pmatrix} \mathtt{P}_{xy_i} \\ \mathtt{P}_{y_1y_i} \\ \mathtt{P}_{y_2y_i} \\ \vdots \end{pmatrix} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top . \end{aligned} \tag{6.8}$$

The innovation covariance $\mathtt{S}$ is given by:

$$\begin{aligned}\mathtt{S} &= \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}}\mathtt{P}\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}}^\top + \mathtt{R} \\ &= \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}\mathtt{P}_{xx}\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top + \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}\mathtt{P}_{y_i x}\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top + \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}\mathtt{P}_{xy_i}\frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top + \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}\mathtt{P}_{y_i y_i}\frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top + \mathtt{R}\,, \end{aligned} \tag{6.9}$$

where $\mathtt{R}$ is the covariance matrix of the measurement noise.

The Kalman gain $\mathtt{W}$ is defined (in Equation 5.6) as:

$$\begin{aligned}\mathtt{W} &= \mathtt{P}\frac{\partial \mathbf{h}}{\partial \mathbf{x}}^\top \mathtt{S}^{-1} \\ &= \begin{pmatrix} \mathtt{P}_{xx} \\ \mathtt{P}_{y_1 x} \\ \mathtt{P}_{y_2 x} \\ \vdots \end{pmatrix} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top \mathtt{S}^{-1} + \begin{pmatrix} \mathtt{P}_{xy_i} \\ \mathtt{P}_{y_1 y_i} \\ \mathtt{P}_{y_2 y_i} \\ \vdots \end{pmatrix} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top \mathtt{S}^{-1}\,. \end{aligned} \tag{6.10}$$

So $\mathtt{WSW}^\top$, which we need as shown in Equation 5.4 to calculate the update in $\mathtt{P}$ after a measurement can be written as

$$\begin{aligned}\mathtt{WSW}^\top &= \left(\mathtt{P}\frac{\partial \mathbf{h}}{\partial \mathbf{x}}^\top \mathtt{S}^{-1}\right)\mathtt{S}\left(\mathtt{P}\frac{\partial \mathbf{h}}{\partial \mathbf{x}}^\top \mathtt{S}^{-1}\right)^\top \\ &= \mathtt{P}\frac{\partial \mathbf{h}}{\partial \mathbf{x}}^\top \mathtt{S}^{-1}\mathtt{S}\mathtt{S}^{-\top}\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\mathtt{P}^\top \\ &= \mathtt{P}\frac{\partial \mathbf{h}}{\partial \mathbf{x}}^\top \mathtt{S}^{-\top}\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\mathtt{P}^\top\,. \end{aligned} \tag{6.11}$$

Substituting for $\mathtt{P}\frac{\partial \mathbf{h}}{\partial \mathbf{x}}^\top$ from Equation 6.8:

$$\begin{aligned}\mathtt{WSW}^\top &= \begin{pmatrix} \mathtt{P}_{xx} \\ \mathtt{P}_{y_1 x} \\ \mathtt{P}_{y_2 x} \\ \vdots \end{pmatrix} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top \mathtt{S}^{-1}\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}\begin{pmatrix} \mathtt{P}_{xx} & \mathtt{P}_{xy_1} & \mathtt{P}_{xy_2} & \dots \end{pmatrix} \\ &+ \begin{pmatrix} \mathtt{P}_{xx} \\ \mathtt{P}_{y_1 x} \\ \mathtt{P}_{y_2 x} \\ \vdots \end{pmatrix} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top \mathtt{S}^{-1}\frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}\begin{pmatrix} \mathtt{P}_{y_i x} & \mathtt{P}_{y_i y_1} & \mathtt{P}_{y_i y_2} & \dots \end{pmatrix} \\ &+ \begin{pmatrix} \mathtt{P}_{xy_i} \\ \mathtt{P}_{y_1 y_i} \\ \mathtt{P}_{y_2 y_i} \\ \vdots \end{pmatrix} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top \mathtt{S}^{-1}\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}\begin{pmatrix} \mathtt{P}_{xx} & \mathtt{P}_{xy_1} & \mathtt{P}_{xy_2} & \dots \end{pmatrix} \\ &+ \begin{pmatrix} \mathtt{P}_{xy_i} \\ \mathtt{P}_{y_1 y_i} \\ \mathtt{P}_{y_2 y_i} \\ \vdots \end{pmatrix} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top \mathtt{S}^{-1}\frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}\begin{pmatrix} \mathtt{P}_{y_i x} & \mathtt{P}_{y_i y_1} & \mathtt{P}_{y_i y_2} & \dots \end{pmatrix}\,. \end{aligned} \tag{6.12}$$

Note that $\mathtt{S}$ is symmetric so $\mathtt{S}^{-\top} = \mathtt{S}^{-1}$.

6.2.3 Updating Particular Parts of the Estimated State Vector and Covariance Matrix

We can analyse the above formulae to determine how particular parts of the estimated state vector and covariance matrix change when a prediction or update occurs. We do this by substituting the expressions for $\mathtt{W}$ and $\mathtt{WSW}^\top$ derived in Equations 6.10 and 6.12 into the general Kalman Filter Equations 5.1–5.4 and observing the changes to particular regions of the estimated state vector and covariance matrix. We assume that the feature observed in the measurement stage has label i, and also consider two general unobserved features j and k.

Firstly, for the estimated state vector $\hat{\mathbf{x}}$, we will determine what happens to the parts of the vector corresponding to a) the sensor state, b) the observed feature and c) an unobserved feature.

- Estimated Sensor State
  Prediction

$$\hat{\mathbf{x}}_{v(new)} = \mathbf{f}_v(\hat{\mathbf{x}}_v, \mathbf{u}) \ . \tag{6.13}$$

  Update

$$\hat{\mathbf{x}}_{v(new)} = \hat{\mathbf{x}}_v + \mathtt{P}_{xx}\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top \mathtt{S}^{-1}\nu + \mathtt{P}_{xy_i}\frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top \mathtt{S}^{-1}\nu \ . \tag{6.14}$$

- Observed Feature State
  Prediction

$$\hat{\mathbf{y}}_{i(new)} = \hat{\mathbf{y}}_i \ . \tag{6.15}$$

  Update

$$\hat{\mathbf{y}}_{i(new)} = \hat{\mathbf{y}}_i + \mathtt{P}_{y_i x}\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top \mathtt{S}^{-1}\nu + \mathtt{P}_{y_i y_i}\frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top \mathtt{S}^{-1}\nu \ . \tag{6.16}$$

- Unobserved Feature State
  Prediction

$$\hat{\mathbf{y}}_{j(new)} = \hat{\mathbf{y}}_j \ . \tag{6.17}$$

  Update

$$\hat{\mathbf{y}}_{j(new)} = \hat{\mathbf{y}}_j + \mathtt{P}_{y_j x}\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top \mathtt{S}^{-1}\nu + \mathtt{P}_{y_j y_i}\frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top \mathtt{S}^{-1}\nu \ . \tag{6.18}$$

Considering now the covariance matrix $\mathtt{P}$, we will determine what happens to the partitions corresponding to the covariance between a) the sensor state and itself, b) the sensor state and the observed feature state, c) the observed feature state and itself, d) the sensor state and an unobserved feature state, e) the observed feature state and an unobserved feature state, and f) two unobserved feature states. These combinations cover all the elements of the covariance matrix.

To simplify the following expressions we define these matrices:

$$\mathtt{A} = \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top \mathtt{S}^{-1} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v} \tag{6.19}$$

$$\mathtt{B} = \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top \mathtt{S}^{-1} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i} \tag{6.20}$$

$$\mathtt{C} = \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top \mathtt{S}^{-1} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v} \tag{6.21}$$

$$\mathtt{D} = \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top \mathtt{S}^{-1} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i} \tag{6.22}$$

- Covariance between Sensor State and Itself
  Prediction

$$\mathtt{P}_{xx(new)} = \frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v} \mathtt{P}_{xx} \frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v}^\top + \mathtt{Q} \ . \tag{6.23}$$

  Update

$$\mathtt{P}_{xx(new)} = \mathtt{P}_{xx} - (\mathtt{P}_{xx}\mathtt{A}\mathtt{P}_{xx} + \mathtt{P}_{xx}\mathtt{B}\mathtt{P}_{y_i x} + \mathtt{P}_{xy_i}\mathtt{C}\mathtt{P}_{xx} + \mathtt{P}_{xy_i}\mathtt{D}\mathtt{P}_{y_i x}) \ . \tag{6.24}$$

- Covariance between Sensor State and Observed Feature State
  Prediction

$$\mathtt{P}_{xy_i(new)} = \frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v} \mathtt{P}_{xy_i} \ . \tag{6.25}$$

  Update

$$\mathtt{P}_{xy_i(new)} = \mathtt{P}_{xy_i} - (\mathtt{P}_{xx}\mathtt{A}\mathtt{P}_{xy_i} + \mathtt{P}_{xx}\mathtt{B}\mathtt{P}_{y_i y_i} + \mathtt{P}_{xy_i}\mathtt{C}\mathtt{P}_{xy_i} + \mathtt{P}_{xy_i}\mathtt{D}\mathtt{P}_{y_i y_i}) \ . \tag{6.26}$$

- Covariance between Observed Feature State and Itself
  Prediction

$$\mathtt{P}_{y_i y_i(new)} = \mathtt{P}_{y_i y_i} \ . \tag{6.27}$$

  Update

$$\mathtt{P}_{y_i y_i(new)} = \mathtt{P}_{y_i y_i} - (\mathtt{P}_{y_i x}\mathtt{A}\mathtt{P}_{xy_i} + \mathtt{P}_{y_i x}\mathtt{B}\mathtt{P}_{y_i y_i} + \mathtt{P}_{y_i y_i}\mathtt{C}\mathtt{P}_{xy_i} + \mathtt{P}_{y_i y_i}\mathtt{D}\mathtt{P}_{y_i y_i}) \ . \tag{6.28}$$

- Covariance between Sensor State and an Unobserved Feature State
  Prediction

$$\mathtt{P}_{xy_j(new)} = \frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v} \mathtt{P}_{xy_j} \ . \tag{6.29}$$

  Update

$$\mathtt{P}_{xy_j(new)} = \mathtt{P}_{xy_j} - \left(\mathtt{P}_{xx}\mathtt{A}\mathtt{P}_{xy_j} + \mathtt{P}_{xx}\mathtt{B}\mathtt{P}_{y_i y_j} + \mathtt{P}_{xy_i}\mathtt{C}\mathtt{P}_{xy_j} + \mathtt{P}_{xy_i}\mathtt{D}\mathtt{P}_{y_i y_j}\right) \ . \tag{6.30}$$

- Covariance between Observed Feature State and an Unobserved Feature State
  Prediction

$$\mathtt{P}_{y_i y_j(new)} = \mathtt{P}_{y_i y_j} \ . \tag{6.31}$$

  Update

$$\mathtt{P}_{y_i y_j(new)} = \mathtt{P}_{y_i y_j} - \left(\mathtt{P}_{y_i x}\mathtt{AP}_{x y_j} + \mathtt{P}_{y_i x}\mathtt{BP}_{y_i y_j} + \mathtt{P}_{y_i y_i}\mathtt{CP}_{x y_j} + \mathtt{P}_{y_i y_i}\mathtt{DP}_{y_i y_j}\right) \ . \tag{6.32}$$

- Covariance between Two Unobserved Feature States
  Prediction

$$\mathtt{P}_{y_j y_k(new)} = \mathtt{P}_{y_j y_k} \ . \tag{6.33}$$

  Update

$$\mathtt{P}_{y_j y_k(new)} = \mathtt{P}_{y_j y_k} - \left(\mathtt{P}_{y_j x}\mathtt{AP}_{x y_k} + \mathtt{P}_{y_j x}\mathtt{BP}_{y_i y_k} + \mathtt{P}_{y_j y_i}\mathtt{CP}_{x y_k} + \mathtt{P}_{y_j y_i}\mathtt{DP}_{y_i y_k}\right) \ . \tag{6.34}$$

  Note that this last expression also covers the case where $j = k$, where it is the covariance between an unobserved feature state and itself.

6.2.4 Multiple Steps

The form of the expressions above allows efficient filtering to be performed in the case where a single feature is observed exclusively over multiple time steps. This of course occurs frequently in the active navigation system described in this thesis where single features are tracked for extended periods.

Let the estimated state vector and covariance at the start of a motion where a single feature i is to be tracked be:

$$\hat{\mathbf{x}}(0) = \begin{pmatrix} \hat{\mathbf{x}}_v(0) \\ \hat{\mathbf{y}}_1(0) \\ \hat{\mathbf{y}}_2(0) \\ \vdots \end{pmatrix} \quad , \quad \mathtt{P}(0) = \begin{bmatrix} \mathtt{P}_{xx}(0) & \mathtt{P}_{xy_1}(0) & \mathtt{P}_{xy_2}(0) & \dots \\ \mathtt{P}_{y_1 x}(0) & \mathtt{P}_{y_1 y_1}(0) & \mathtt{P}_{y_1 y_2}(0) & \dots \\ \mathtt{P}_{y_2 x}(0) & \mathtt{P}_{y_2 y_1}(0) & \mathtt{P}_{y_2 y_2}(0) & \dots \\ \vdots & \vdots & \vdots & \end{bmatrix} \ . \tag{6.35}$$

During the extended period of measurements of one feature, Equations 6.13–6.16 and 6.23–6.28 show that it will be necessary to update the parts of the state vector and covariance matrix which are directly involved in the tracking process at all times: these are the estimated states of the sensor and observed feature, $\hat{\mathbf{x}}_v$ and $\hat{\mathbf{y}}_i$, and the covariance elements $\mathtt{P}_{xx}$, $\mathtt{P}_{xy_i}$ and $\mathtt{P}_{y_i y_i}$. However, it will not be essential to update the other parts of the estimated state vector and covariance matrix at every step: they play no part in the updates of the quantities just mentioned. We will show how a small amount of information can be stored at each filter step which allows the other sections of the state and covariance to be updated in a generic way at the end of the tracking motion.

- $\mathtt{P}_{xy_j}$ and $\mathtt{P}_{y_i y_j}$

We look first at the parts $\mathtt{P}_{xy_j}$ and $\mathtt{P}_{y_iy_j}$ of the covariance matrix: the forms of their prediction and update changes given in Equations 6.29–6.32 suggest that it is possible always to express them in the form

$$\mathtt{P}_{xy_j} = \mathtt{E}_T\mathtt{P}_{xy_j}(0) + \mathtt{F}_T\mathtt{P}_{y_iy_j}(0) \tag{6.36}$$
$$\mathtt{P}_{y_iy_j} = \mathtt{G}_T\mathtt{P}_{y_iy_j}(0) + \mathtt{H}_T\mathtt{P}_{xy_j}(0)\ , \tag{6.37}$$

where $\mathtt{E}_T$, $\mathtt{F}_T$, $\mathtt{G}_T$ and $\mathtt{H}_T$ are as yet undetermined matrices. To find how these matrices evolve over time, we consider first a single measurement update. From Equations 6.30 and 6.32:

$$\mathtt{P}_{xy_j(new)} = \mathtt{P}_{xy_j} - \left[\mathtt{P}_{xx}\mathtt{A} + \mathtt{P}_{xy_i}\mathtt{C}\right]\mathtt{P}_{xy_j} - \left[\mathtt{P}_{xx}\mathtt{B} + \mathtt{P}_{xy_i}\mathtt{D}\right]\mathtt{P}_{y_iy_j}\ , \tag{6.38}$$
$$\mathtt{P}_{y_iy_j(new)} = \mathtt{P}_{y_iy_j} - \left[\mathtt{P}_{y_ix}\mathtt{A} + \mathtt{P}_{y_iy_i}\mathtt{C}\right]\mathtt{P}_{xy_j} - \left[\mathtt{P}_{y_ix}\mathtt{B} + \mathtt{P}_{y_iy_i}\mathtt{D}\right]\mathtt{P}_{y_iy_j}\ . \tag{6.39}$$
$$\tag{6.40}$$

We define the matrices $\mathtt{E}$, $\mathtt{F}$, $\mathtt{G}$ and $\mathtt{H}$ as:

$$\mathtt{E} = \mathtt{I} - \left[\mathtt{P}_{xx}\mathtt{A} + \mathtt{P}_{xy_i}\mathtt{C}\right] \tag{6.41}$$
$$\mathtt{F} = -\left[\mathtt{P}_{xx}\mathtt{B} + \mathtt{P}_{xy_i}\mathtt{D}\right] \tag{6.42}$$
$$\mathtt{G} = \mathtt{I} - \left[\mathtt{P}_{y_ix}\mathtt{B} + \mathtt{P}_{y_iy_i}\mathtt{D}\right] \tag{6.43}$$
$$\mathtt{H} = -\left[\mathtt{P}_{y_ix}\mathtt{A} + \mathtt{P}_{y_iy_i}\mathtt{C}\right] \tag{6.44}$$

So:

$$\mathtt{P}_{xy_j(new)} = \mathtt{E}\mathtt{P}_{xy_j} + \mathtt{F}\mathtt{P}_{y_iy_j}\ , \tag{6.45}$$
$$\mathtt{P}_{y_iy_j(new)} = \mathtt{G}\mathtt{P}_{y_iy_j} + \mathtt{H}\mathtt{P}_{xy_j}\ . \tag{6.46}$$

Therefore, recalling the form in which we would like to express $\mathtt{P}_{xy_j}$ and $\mathtt{P}_{y_iy_j}$ given in Equations 6.36 and 6.37, the coefficient matrices $\mathtt{E}_T$, $\mathtt{F}_T$, $\mathtt{G}_T$ and $\mathtt{H}_T$ should be altered as follows:

$$\mathtt{E}_{T(new)} = \mathtt{E}\mathtt{E}_T + \mathtt{F}\mathtt{H}_T \tag{6.47}$$
$$\mathtt{F}_{T(new)} = \mathtt{E}\mathtt{F}_T + \mathtt{F}\mathtt{G}_T \tag{6.48}$$
$$\mathtt{G}_{T(new)} = \mathtt{G}\mathtt{G}_T + \mathtt{H}\mathtt{F}_T \tag{6.49}$$
$$\mathtt{H}_{T(new)} = \mathtt{G}\mathtt{H}_T + \mathtt{H}\mathtt{E}_T \tag{6.50}$$

in a measurement update.

Altering $\mathtt{E}_T$, $\mathtt{F}_T$, $\mathtt{G}_T$ and $\mathtt{H}_T$ at a prediction is much easier: Equations 6.29 and 6.31 tell us that:

$$\mathtt{E}_{T(new)} = \frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v}\mathtt{E}_T\ , \tag{6.51}$$
$$\mathtt{F}_{T(new)} = \frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v}\mathtt{F}_T\ . \tag{6.52}$$

$\mathtt{G}_T$ and $\mathtt{H}_T$ are unchanged.

- $\mathtt{P}_{y_j y_k}$

  With this part of the covariance matrix, the covariance between the position estimate of two of the unobserved features, the prediction and measurement updates in Equations 6.33 and 6.34 suggest that we can express it in the form:

$$\begin{aligned}\mathtt{P}_{y_j y_k} = \mathtt{P}_{y_j y_k}(0) - [\quad & \mathtt{P}_{y_j x}(0)\mathtt{A}_T\mathtt{P}_{x y_k}(0) \\ + \quad & \mathtt{P}_{y_j x}(0)\mathtt{B}_T\mathtt{P}_{y_i y_k}(0) \\ + \quad & \mathtt{P}_{y_j y_i}(0)\mathtt{C}_T\mathtt{P}_{x y_k}(0) \\ + \quad & \mathtt{P}_{y_j y_i}(0)\mathtt{D}_T\mathtt{P}_{y_i y_k}(0) \quad] \, . \end{aligned} \tag{6.53}$$

  From Equation 6.34 we can write down the form of a single measurement update:

$$\mathtt{P}_{y_j y_k(new)} = \mathtt{P}_{y_j y_k} - [\mathtt{P}_{y_j x}\mathtt{A}\mathtt{P}_{x y_k} + \mathtt{P}_{y_j x}\mathtt{B}\mathtt{P}_{y_i y_k} + \mathtt{P}_{y_j y_i}\mathtt{C}\mathtt{P}_{x y_k} + \mathtt{P}_{y_j y_i}\mathtt{D}\mathtt{P}_{y_i y_k}] \tag{6.54}$$

  Substituting in for $\mathtt{P}_{x y_j}$, $\mathtt{P}_{y_i y_j}$, $\mathtt{P}_{x y_k}$ and $\mathtt{P}_{y_i y_k}$ (and their transposes) from the expressions found in the previous section (Equations 6.36 and 6.37) and rearranging, we obtain:

$$\begin{aligned}\mathtt{P}_{y_j y_k(new)} \quad = \quad & \mathtt{P}_{y_j y_k} \\ - \quad & \mathtt{P}_{y_j x}(0)[\mathtt{E}_T^\top\mathtt{A}\mathtt{E}_T + \mathtt{E}_T^\top\mathtt{B}\mathtt{H}_T + \mathtt{H}_T^\top\mathtt{C}\mathtt{E}_T + \mathtt{H}_T^\top\mathtt{D}\mathtt{H}_T]\mathtt{P}_{x y_k}(0) \\ - \quad & \mathtt{P}_{y_j x}(0)[\mathtt{E}_T^\top\mathtt{A}\mathtt{F}_T + \mathtt{E}_T^\top\mathtt{B}\mathtt{G}_T + \mathtt{H}_T^\top\mathtt{C}\mathtt{F}_T + \mathtt{H}_T^\top\mathtt{D}\mathtt{G}_T]\mathtt{P}_{y_i y_k}(0) \\ - \quad & \mathtt{P}_{y_j y_i}(0)[\mathtt{F}_T^\top\mathtt{A}\mathtt{E}_T + \mathtt{F}_T^\top\mathtt{B}\mathtt{H}_T + \mathtt{G}_T^\top\mathtt{C}\mathtt{E}_T + \mathtt{G}_T^\top\mathtt{D}\mathtt{H}_T]\mathtt{P}_{x y_k}(0) \\ - \quad & \mathtt{P}_{y_j y_i}(0)[\mathtt{F}_T^\top\mathtt{A}\mathtt{F}_T + \mathtt{F}_T^\top\mathtt{B}\mathtt{G}_T + \mathtt{G}_T^\top\mathtt{C}\mathtt{F}_T + \mathtt{G}_T^\top\mathtt{D}\mathtt{G}_T]\mathtt{P}_{y_i y_k}(0) \end{aligned} \tag{6.55}$$

  We can therefore deduce the way in which the coefficient matrices $\mathtt{A}_T$, $\mathtt{B}_T$, $\mathtt{C}_T$ and $\mathtt{D}_T$ should be altered at a measurement update:

$$\mathtt{A}_{T(new)} = \mathtt{A}_T + \mathtt{E}_T^\top\mathtt{A}\mathtt{E}_T + \mathtt{E}_T^\top\mathtt{B}\mathtt{H}_T + \mathtt{H}_T^\top\mathtt{C}\mathtt{E}_T + \mathtt{H}_T^\top\mathtt{D}\mathtt{H}_T \tag{6.56}$$

$$\mathtt{B}_{T(new)} = \mathtt{B}_T + \mathtt{E}_T^\top\mathtt{A}\mathtt{F}_T + \mathtt{E}_T^\top\mathtt{B}\mathtt{G}_T + \mathtt{H}_T^\top\mathtt{C}\mathtt{F}_T + \mathtt{H}_T^\top\mathtt{D}\mathtt{G}_T \tag{6.57}$$

$$\mathtt{C}_{T(new)} = \mathtt{C}_T + \mathtt{F}_T^\top\mathtt{A}\mathtt{E}_T + \mathtt{F}_T^\top\mathtt{B}\mathtt{H}_T + \mathtt{G}_T^\top\mathtt{C}\mathtt{E}_T + \mathtt{G}_T^\top\mathtt{D}\mathtt{H}_T \tag{6.58}$$

$$\mathtt{D}_{T(new)} = \mathtt{D}_T + \mathtt{F}_T^\top\mathtt{A}\mathtt{F}_T + \mathtt{F}_T^\top\mathtt{B}\mathtt{G}_T + \mathtt{G}_T^\top\mathtt{C}\mathtt{F}_T + \mathtt{G}_T^\top\mathtt{D}\mathtt{G}_T \tag{6.59}$$

  Since Equation 6.33 shows that $\mathtt{P}_{y_j y_k}$ is unchanged in a prediction, it is not necessary to change $\mathtt{A}_T$, $\mathtt{B}_T$, $\mathtt{C}_T$ and $\mathtt{D}_T$ when a prediction occurs.

- $\hat{\mathbf{y}}_j$

  With the estimated state $\hat{\mathbf{y}}_j$ of an unobserved feature, looking at Equations 6.17 and 6.18 suggests that we can write it as

$$\hat{\mathbf{y}}_j = \hat{\mathbf{y}}_j(0) + \mathtt{P}_{y_j x}(0)\mathbf{m}_T + \mathtt{P}_{y_j y_i}(0)\mathbf{n}_T \ , \tag{6.60}$$

  where $\mathbf{m}_T$ and $\mathbf{n}_T$ are vectors whose form is to be determined.

  From Equation 6.18, the form of a measurement update is

$$\hat{\mathbf{y}}_{j(new)} = \hat{\mathbf{y}}_j + \mathtt{P}_{y_j x}\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top \mathtt{S}^{-1}\nu + \mathtt{P}_{y_j y_i}\frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top \mathtt{S}^{-1}\nu \ . \tag{6.61}$$

We substitute in the expressions for $\mathtt{P}_{y_j x}$ and $\mathtt{P}_{y_j y_i}$ from Equations 6.36 and 6.37 to obtain:

$$\begin{aligned}\hat{\mathbf{y}}_{j(new)} &= \hat{\mathbf{y}}_j \\ &\quad +\mathtt{P}_{y_j x}(0)\left(\mathtt{E}_T^\top \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top + \mathtt{H}_T^\top \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top\right)\mathtt{S}^{-1}\nu \\ &\quad +\mathtt{P}_{y_j y_i}(0)\left(\mathtt{F}_T^\top \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top + \mathtt{G}_T^\top \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top\right)\mathtt{S}^{-1}\nu\end{aligned} \tag{6.62}$$

Therefore, in a measurement step we alter $\mathbf{m}_T$ and $\mathbf{n}_T$ as follows:

$$\mathbf{m}_{T(new)} = \mathbf{m}_T + \left(\mathtt{E}_T^\top \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top + \mathtt{H}_T^\top \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top\right)\mathtt{S}^{-1}\nu \tag{6.63}$$

$$\mathbf{n}_{T(new)} = \mathbf{n}_T + \left(\mathtt{F}_T^\top \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top + \mathtt{G}_T^\top \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top\right)\mathtt{S}^{-1}\nu \tag{6.64}$$

In a prediction step, Equation 6.17 shows that we do not need to alter $\mathbf{m}_T$ and $\mathbf{n}_T$ because $\hat{\mathbf{y}}_j$ does not change.

6.2.5 Summary

For clarity, we summarise the method described above. The situation under consideration is that of a sensor moving amongst multiple features of interest, but currently making measurements of only one, labelled i. The labels j and k describe two general unobserved features. The method describes the way in which, while the parts of the estimated state vector and covariance matrix which are directly involved in the motion and measurements are continuously updated, the information about how generically to update the other sections can be efficiently stored so that a single step can bring them up to date at the end of the motion.

At the start of the motion, the estimated state vector and covariance of the system are denoted by $\hat{\mathbf{x}}(0)$ and $\mathtt{P}(0)$. The following matrices and vectors, which will store the generic update information (the T subscript being coined with the word "total" in mind), are initialised as follows:

$$\begin{aligned}\mathtt{A}_T &= \mathtt{0} \\ \mathtt{B}_T &= \mathtt{0} \\ \mathtt{C}_T &= \mathtt{0} \\ \mathtt{D}_T &= \mathtt{0} \\ \mathtt{E}_T &= \mathtt{I} \\ \mathtt{F}_T &= \mathtt{0} \\ \mathtt{G}_T &= \mathtt{I} \\ \mathtt{H}_T &= \mathtt{0} \\ \mathbf{m}_T &= \mathbf{0} \\ \mathbf{n}_T &= \mathbf{0}\end{aligned} \tag{6.65}$$

The matrices $\mathtt{A}_T \ldots \mathtt{H}_T$ are square with dimension equal to the size of the vector $\mathbf{y}_j$ describing the position of a feature. $\mathbf{m}_T$ and $\mathbf{n}_T$ are column vectors of the same dimension.

Prediction

When the sensor moves, the following parts of the estimated state vector and covariance should be directly updated as follows:

$$\hat{\mathbf{x}}_{v(new)} = \mathbf{f}_v(\hat{\mathbf{x}}_v, \mathbf{u}) \tag{6.66}$$

$$\hat{\mathbf{y}}_{i(new)} = \hat{\mathbf{y}}_i \tag{6.67}$$

$$\mathtt{P}_{xx(new)} = \frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v} \mathtt{P}_{xx} \frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v}^\top + \mathtt{Q} \tag{6.68}$$

$$\mathtt{P}_{xy_i(new)} = \frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v} P_{xy_i} \tag{6.69}$$

$$\mathtt{P}_{y_i y_i(new)} = P_{y_i y_i} \tag{6.70}$$

The following changes should be made to the stored matrices:

$$\mathtt{E}_{T(new)} = \frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v} \mathtt{E}_T \tag{6.71}$$

$$\mathtt{F}_{T(new)} = \frac{\partial \mathbf{f}_v}{\partial \mathbf{x}_v} \mathtt{F}_T \tag{6.72}$$

Measurement Update

When a measurement of feature i is obtained, these matrices should be calculated:

$$\mathtt{A} = \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top \mathtt{S}^{-1} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v} \tag{6.73}$$

$$\mathtt{B} = \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top \mathtt{S}^{-1} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i} \tag{6.74}$$

$$\mathtt{C} = \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top \mathtt{S}^{-1} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v} \tag{6.75}$$

$$\mathtt{D} = \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top \mathtt{S}^{-1} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i} \tag{6.76}$$

$$\mathtt{E} = \mathtt{I} - [\mathtt{P}_{xx}\mathtt{A} + \mathtt{P}_{xy_i}\mathtt{C}] \tag{6.77}$$

$$\mathtt{F} = -[\mathtt{P}_{xx}\mathtt{B} + \mathtt{P}_{xy_i}\mathtt{D}] \tag{6.78}$$

$$\mathtt{G} = \mathtt{I} - [\mathtt{P}_{y_i x}\mathtt{B} + \mathtt{P}_{y_i y_i}\mathtt{D}] \tag{6.79}$$

$$\mathtt{H} = -[\mathtt{P}_{y_i x}\mathtt{A} + \mathtt{P}_{y_i y_i}\mathtt{C}] \tag{6.80}$$

The following direct updates are made:

$$\hat{\mathbf{x}}_{v(new)} = \hat{\mathbf{x}}_v + \mathtt{P}_{xx} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top \mathtt{S}^{-1}\nu + \mathtt{P}_{xy_i} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top \mathtt{S}^{-1}\nu \tag{6.81}$$

$$\hat{\mathbf{y}}_{i(new)} = \hat{\mathbf{y}}_i + \mathtt{P}_{y_i x} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top \mathtt{S}^{-1}\nu + \mathtt{P}_{y_i y_i} \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top \mathtt{S}^{-1}\nu \tag{6.82}$$

$$\mathtt{P}_{xx(new)} = \mathtt{P}_{xx} - (\mathtt{P}_{xx}\mathtt{A}\mathtt{P}_{xx} + \mathtt{P}_{xx}\mathtt{B}\mathtt{P}_{y_ix} + \mathtt{P}_{xy_i}\mathtt{C}\mathtt{P}_{xx} + \mathtt{P}_{xy_i}\mathtt{D}\mathtt{P}_{y_ix}) \quad (6.83)$$

$$\mathtt{P}_{xy_i(new)} = \mathtt{P}_{xy_i} - (\mathtt{P}_{xx}\mathtt{A}\mathtt{P}_{xy_i} + \mathtt{P}_{xx}\mathtt{B}\mathtt{P}_{y_iy_i} + \mathtt{P}_{xy_i}\mathtt{C}\mathtt{P}_{xy_i} + \mathtt{P}_{xy_i}\mathtt{D}\mathtt{P}_{y_iy_i}) \quad (6.84)$$

$$\mathtt{P}_{y_iy_i(new)} = \mathtt{P}_{y_iy_i} - (\mathtt{P}_{y_ix}\mathtt{A}\mathtt{P}_{xy_i} + \mathtt{P}_{y_ix}\mathtt{B}\mathtt{P}_{y_iy_i} + \mathtt{P}_{y_iy_i}\mathtt{C}\mathtt{P}_{xy_i} + \mathtt{P}_{y_iy_i}\mathtt{D}\mathtt{P}_{y_iy_i}) \quad (6.85)$$

These changes are then made to the stored matrices and vectors:

$$\mathbf{m}_{T(new)} = \mathbf{m}_T + \left(\mathtt{E}_T^\top \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top + \mathtt{H}_T^\top \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top\right) \mathtt{S}^{-1}\nu \quad (6.86)$$

$$\mathbf{n}_{T(new)} = \mathbf{n}_T + \left(\mathtt{F}_T^\top \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_v}^\top + \mathtt{G}_T^\top \frac{\partial \mathbf{h}_i}{\partial \mathbf{y}_i}^\top\right) \mathtt{S}^{-1}\nu \quad (6.87)$$

$$\mathtt{A}_{T(new)} = \mathtt{A}_T + \mathtt{E}_T^\top\mathtt{A}\mathtt{E}_T + \mathtt{E}_T^\top\mathtt{B}\mathtt{H}_T + \mathtt{H}_T^\top\mathtt{C}\mathtt{E}_T + \mathtt{H}_T^\top\mathtt{D}\mathtt{H}_T \quad (6.88)$$

$$\mathtt{B}_{T(new)} = \mathtt{B}_T + \mathtt{E}_T^\top\mathtt{A}\mathtt{F}_T + \mathtt{E}_T^\top\mathtt{B}\mathtt{G}_T + \mathtt{H}_T^\top\mathtt{C}\mathtt{F}_T + \mathtt{H}_T^\top\mathtt{D}\mathtt{G}_T \quad (6.89)$$

$$\mathtt{C}_{T(new)} = \mathtt{C}_T + \mathtt{F}_T^\top\mathtt{A}\mathtt{E}_T + \mathtt{F}_T^\top\mathtt{B}\mathtt{H}_T + \mathtt{G}_T^\top\mathtt{C}\mathtt{E}_T + \mathtt{G}_T^\top\mathtt{D}\mathtt{H}_T \quad (6.90)$$

$$\mathtt{D}_{T(new)} = \mathtt{D}_T + \mathtt{F}_T^\top\mathtt{A}\mathtt{F}_T + \mathtt{F}_T^\top\mathtt{B}\mathtt{G}_T + \mathtt{G}_T^\top\mathtt{C}\mathtt{F}_T + \mathtt{G}_T^\top\mathtt{D}\mathtt{G}_T \quad (6.91)$$

$$\mathtt{E}_{T(new)} = \mathtt{E}\mathtt{E}_T + \mathtt{F}\mathtt{H}_T \quad (6.92)$$

$$\mathtt{F}_{T(new)} = \mathtt{E}\mathtt{F}_T + \mathtt{F}\mathtt{G}_T \quad (6.93)$$

$$\mathtt{G}_{T(new)} = \mathtt{G}\mathtt{G}_T + \mathtt{H}\mathtt{F}_T \quad (6.94)$$

$$\mathtt{H}_{T(new)} = \mathtt{G}\mathtt{H}_T + \mathtt{H}\mathtt{E}_T \quad (6.95)$$

Note that the order of these changes is important because the expressions updating $\mathbf{m}_T$, $\mathbf{n}_T$ and $\mathtt{A}_T \ldots \mathtt{D}_T$ make use of the old values of $\mathtt{E}_T \ldots \mathtt{H}_T$. When performing the updates of $\mathtt{E}_T \ldots \mathtt{H}_T$ in Equations 6.92 – 6.95, the matrices on the right hand side of the expressions always refer to the old versions of $\mathtt{E}_T \ldots \mathtt{H}_T$, so care must be taken here and some temporary storage used.

So the computational expense involved in making updates to the stored matrices and vectors at a prediction or measurement is *constant*, independent of the number of features in the map. We store just information on the generic way in which the currently unimportant parts of the total state vector and covariance matrix can later be brought up to date.

Final Full Update

At the end of the motion, the whole estimated state vector can be brought up to date using the following expression for each unobserved feature j:

$$\hat{\mathbf{y}}_j = \hat{\mathbf{y}}_j(0) + \mathtt{P}_{y_jx}(0)\mathbf{m}_T + \mathtt{P}_{y_jy_i}(0)\mathbf{n}_T \ . \quad (6.96)$$

This expression describes how to update the covariance between a pair of unobserved features j and k:

$$\begin{aligned}\mathtt{P}_{y_jy_k} = \mathtt{P}_{y_jy_k}(0) - [\quad & \mathtt{P}_{y_jx}(0)\mathtt{A}_T\mathtt{P}_{xy_k}(0) \\ + \quad & \mathtt{P}_{y_jx}(0)\mathtt{B}_T\mathtt{P}_{y_iy_k}(0) \\ + \quad & \mathtt{P}_{y_jy_i}(0)\mathtt{C}_T\mathtt{P}_{xy_k}(0) \\ + \quad & \mathtt{P}_{y_jy_i}(0)\mathtt{D}_T\mathtt{P}_{y_iy_k}(0) \quad] \ . \end{aligned} \quad (6.97)$$

Finally, the parts of the covariance matrix relating unobserved feature j to the sensor state and observed feature state can be brought up to date as follows:

$$\mathtt{P}_{xy_j} = \mathtt{E}_T \mathtt{P}_{xy_j}(0) + \mathtt{F}_T \mathtt{P}_{y_i y_j}(0) \ , \tag{6.98}$$

$$\mathtt{P}_{y_i y_j} = \mathtt{G}_T \mathtt{P}_{y_i y_j}(0) + \mathtt{H}_T \mathtt{P}_{x y_j}(0) \ . \tag{6.99}$$

Note that the order of operation is again important here, since it is necessary that $\mathtt{P}_{xy_j}$, $\mathtt{P}_{y_i y_j}$, $\mathtt{P}_{xy_k}$ and $\mathtt{P}_{y_i y_k}$ remain at their initial (0) values for the update of the $\mathtt{P}_{y_j y_k}$ matrix element.

This final update stage has a computational complexity which is proportional to the square of the number of features in the map.

6.3 A Strategy for Fixation: Selecting Which Feature to Track

The previous two sections have shown how it is possible to track a feature while the robot is moving to give continuous information on the robot and feature positions. The active head allows tracked features to lie within a very wide field of view, meaning that the same one can be tracked as the robot makes large movements, or alternatively that a variety of different features can be tracked at different stages of a motion. The fact that choice is available means that a strategy for selecting features to be observed is required. As discussed in Section 4.1.2, only a subset of features in the map will be viewable from a certain robot location, due to constraints of viewing direction and distance, but which should be observed at a given time? Also, when is it time to look for new landmarks, and is it sensible to look for them in a particular place?

First, it is necessary to consider in which senses one feature could be "better" than another to make measurements from:

- If measurements of that feature would provide more useful information.
- If less effort is required to make the measurements.

That some measurements can be more valuable than others in terms of information content is quickly apparent: depending on which components of the current estimated quantities, in particular the robot's estimated position coordinates $\hat{z}$, $\hat{x}$ and $\hat{\phi}$, are currently the least certain, it will be more useful to make measurements of features lying in certain positions. If for instance the robot's location is uncertain primarily in the forward/backward direction, there will be little use in making a measurement of a feature which lies directly ahead at a large range, since the large depth uncertainty of this measurement would provide little extra constraint on the robot position.

The factor of the "effort" needed to make a measurement comes into effect when several measurements are being strung together in a certain time interval: time spent moving to observe or searching for one feature could alternatively be used to measure another one. The most clear example is if one feature is currently being tracked at fixation, but there is the possibility of making a saccade to observe another feature while the robot keeps moving: making that potentially large head movement will be costly compared to making the small adjustment to stay fixated on the current feature, so we would expect the information quality

of the alternative feature to have to be greater than that of the current by a reasonable amount to make the saccade worthwhile. This is the question which will be addressed in Section 6.3.3.

Active fixation switching is very evident in human and animal navigation — the human eye is rarely at rest as it rapidly shifts between points of interest. During a period of walking, for example, attention will be divided between the ground surface, obstacles, near and far landmarks and other objects of interest. Blind navigation is unreliable and dangerous, as is easily demonstrated by the discomfort felt during an attempt at walking with closed eyes: the urge to re-open the eyes and reorient is very strong. Further, though, staring at just one feature in the world for an extended period while moving can be disorientating — it is necessary to pay attention to a variety of landmarks to be fully confident of safe navigation. Investigating feature selection in this section casts some light on these natural behaviours as well as providing a key part of the robot navigation system, where the full statistical information stored about the estimates of robot and feature positions means that we are in a position to make reasoned decisions about which features to measure at a given time.

6.3.1 Choosing from Known Features

We will first consider making a choice between known and currently visible features for an immediate measurement. Because it is intended for long-term use, what is important in the map-building system is the integrity and consistency of the whole map and the estimate of the robot's position within it. We aim to maximise this with our choice of feature, rather than specifically the estimate of the robot position or that of the chosen feature — these are quantities which live in the world coordinate frame, which is arbitrarily chosen. The relative locations of robot and features are what matter for navigation. Furthermore, these estimates are all intricately coupled by previous measurements — it would be somewhat meaningless to try to optimise just a subset of them. However, we will see below that choice of feature on this basis usually does provide the best improvement in the explicit world-frame estimate of the robot location.

The basis of the approach used is to make a measurement where the ability to predict is lowest, as discussed and derived in recent work by Whaite and Ferrie [100] in their paper about active exploration to determine the shapes of surfaces. Speaking generally, given a choice of measurements in a system where the uncertainty in estimates of the parameters of interest is known, it makes sense to make the one where we are least certain of the result, since this will in a sense "squash" the total uncertainty, viewed as a multi-dimensional ellipsoid, along the longest axis available. If there is any part of the estimated state vector which is particularly poorly known, this choice of measurement will act to reduce that uncertainty, smoothing off the estimation's rough edges.

In our system, whenever the robot is to make a measurement of a feature, a predicted measurement $\mathbf{h}$ is formed and the innovation covariance matrix $\mathtt{S}$ is calculated as in Section 5.3.4. This matrix describes how much the actual measurement is expected to vary from the prediction in the α, e, γ angular measurement coordinates. The size and shape of the ellipsoid represented by $\mathtt{S}$ will reflect the amount of uncertainty in the estimated relative position of the feature and the robot; measurement noise is constant in this coordinate frame so it only provides a constant addition.

To produce scalar numbers to use as the basis for decisions about which feature to

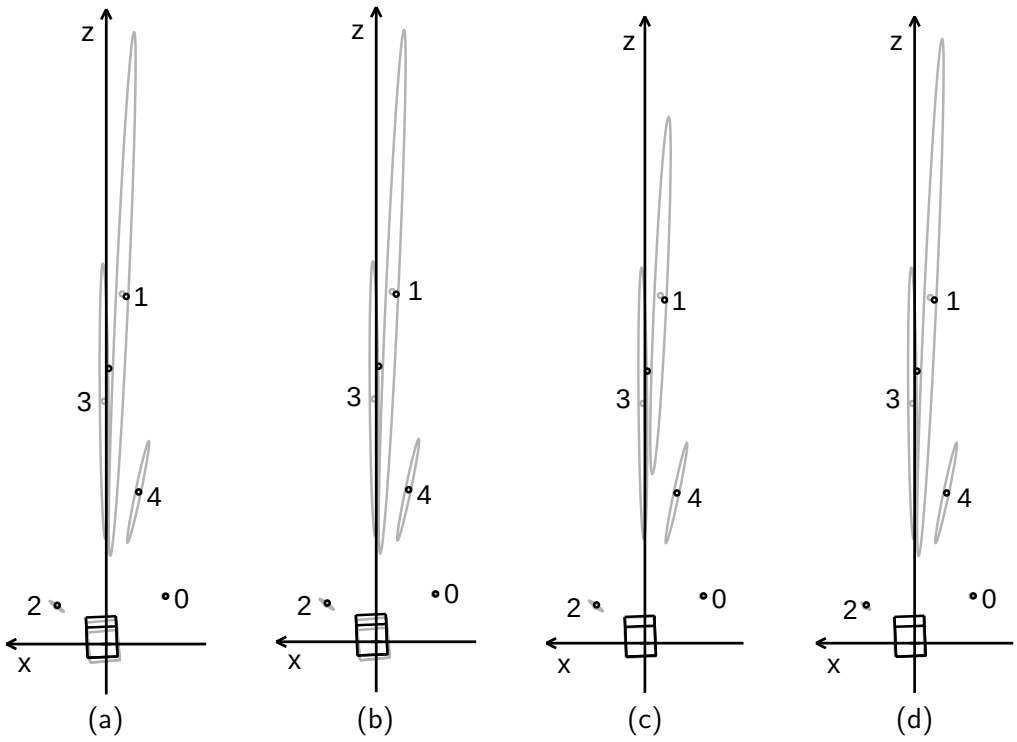


Figure 6.2: Selecting between features after a long period tracking one: in (a) the robot stops after tracking feature 0. In (b), (c) and (d), the estimated state is updated after further measurements of features 0, 1 and 2 respectively. The large improvement in the estimated robot state in (c) and (d) shows the value of making measurements of multiple features.

observe, the volume V_S in (α, e, γ) space of the ellipsoid represented by $\mathtt{S}$ at the $n_\sigma\sigma$ level (again, $n_\sigma = 3$ is normally used) can be calculated for each visible feature: eigenvalues λ_1, λ_2 and λ_3 of $\mathtt{S}$ mean that the ellipsoid has axes of length $n_\sigma\sqrt{\lambda_1}$, $n_\sigma\sqrt{\lambda_2}$ and $n_\sigma\sqrt{\lambda_3}$; so

$$V_S = \frac{4}{3}\pi n_\sigma^3 \sqrt{\lambda_1\lambda_2\lambda_3}\ . \tag{6.100}$$

This quantity is evaluated for each of the features which is currently visible, and the one with the highest value is selected for measurements.

Experiments

The most striking consequence of this criterion seen in experiments is that it demands frequent changes of measured feature. Once a few measurements have been made of a particular one, the criterion tells the robot that there is not much more information to be gleaned from it at the current time, and it is best to switch attention to another. This is a result of the way that tracking one point feature, even with perfect measurements, does not fully constrain the robot's motion — uncertainty is always growing in some direction (recall Figure 5.7).

The following experiment shows this clearly: five widely-spaced features were initialised by the robot from its starting point at the origin. One was arbitrarily selected, and tracked continuously while the robot made a forward and backward motion of around 2m, ending up about 30cm in front of where it started. The situation at the end of this motion is displayed in Figure 6.2(a), where it is feature 0 which has been tracked. The five features were then evaluated according to the criterion above — the values of the innovation covariance volume V_S for each were:

0. $V_S = 0.00004$
1. $V_S = 0.00046$
2. $V_S = 0.00127$
3. $V_S = 0.00049$
4. $V_S = 0.00040$

The clear conclusion is that there is little merit in making yet another measurement of feature 0. Of the alternative features, 2 is the best, with the others being roughly equal.

Figure 6.2(b, c, d) show the estimated state after an extra measurement of features 0, 1 and 2 respectively. The conclusion of the feature choice criterion proves to be successful: making the extra measurement of feature 0 in (b) does little to improve the robot position estimation, which has drifted along the direction ambiguous to measurements of that feature; (c) and (d), however, show the robot estimate locked back onto the ground-truth. There is little difference to be seen between these latter two from the diagrams or between their estimate of the robot state (both of which are within a centimetre or two of the truth); the robot state covariances in the two cases, however, are:

$$\mathtt{P}_{xx}(1) = \begin{bmatrix} 0.00035 & 0.00008 & -0.00013 \\ 0.00008 & 0.00024 & -0.00009 \\ -0.00013 & -0.00009 & 0.00010 \end{bmatrix}$$

$$\mathtt{P}_{xx}(2) = \begin{bmatrix} 0.00010 & 0.00005 & -0.00003 \\ 0.00005 & 0.00021 & -0.00010 \\ -0.00003 & -0.00010 & 0.00009 \end{bmatrix}$$

The second, after a measurement of feature 2 which was favoured by the V_S criterion, is noticeably smaller, showing better confidence in the new robot state estimate. Feature 2 lies much closer to the robot than 1, 3 and 4: compared to these, just the small change of viewpoint in the robot motion provides more chance to look at 2 from a different angle and get some different information than provided from the initial measurement.

6.3.2 Choosing New Features

When it comes to initialising new landmarks, the process of making choices is less clear: it is not as easy to make decisions between features which are not yet known as between those whose positions have been estimated already.

As will be discussed in Chapter 7, the key to maintaining a useful map and always being confident of the robot's position is to find features to track from wherever the robot might be. In most environments encountered, there might be relatively few objects that will serve as reliable landmarks. Therefore, searching for new features lying in some kind of

"optimal" positions relative to the robot might prove futile. The approach we have taken is to consider how to make a choice from an arbitrary set of features which have been newly initialised wherever they could be found.

A fact which is initially slightly surprising, but clear when given consideration, is that the value of V_S for an immediate re-measurement, before the robot has moved, of a feature which has just been initialised is always the same, independent of the location of the feature:

$$V_{S(\text{new})} = \frac{4}{3}\pi(\sqrt{2}n_\sigma)^3\Delta\alpha\Delta e\Delta\gamma \ .$$

This value is 6.91×10^{-5} with the parameters used in our system at the time of experiments (number of standard deviations $n_\sigma = 3$, estimates of angular measurement errors $\Delta\alpha = \Delta e = \Delta\gamma = 0.006$). The reason for this is that the uncertainty in the position relative to the robot of a newly initialised feature is simply the uncertainty in the initialisation measurement — and so the innovation covariance when a second measurement is made of it is the sum of this and the uncertainty in the new measurement: the total is simply double the single measurement uncertainty. In the normalised (α, e, γ) measurement representation this amount is constant. So, despite the fact that the uncertainties in (x, y, z) space look dramatically different for newly initialised features which are close to or far from the robot, in angular space they are all the same. Consider a feature which is very far from the robot, for instance: it will have a large depth uncertainty, represented by an ellipse which is elongated in that direction. The robot can, however, fixate any point along the depth axis with only small changes in its vergence angle — the same range of changes which span the much smaller depth uncertainty range of a close feature.

This means that from a selection of newly initialised features, it will not be possible to choose one to measure first on the same basis as choosing between well-established features: there will not be one feature which as it stands will represent a better measurement than the others. It is clear, however, that once the robot has made a movement, the fact that it has the potential to observe the features from a different viewpoint means that V_S will not be the same in each case. Its value will change depending on how much the view of the feature is different from the original initialising view. We can think of a new measurement uncertainty ellipse in (x, y, z) space intersecting the feature position uncertainty: the measurement will be particularly valuable if the volume mutually enclosed by both is much smaller than the volume of the original ellipse. An extreme case would be two elongated ellipses crossing at right angles: the feature position could then be pinpointed very precisely.

So choice between new features depends on some knowledge of the robot's future motion plan. In our system, the current velocity and steering angle settings and a "number of steps" setting determine the robot's next motion. Based on these, a prediction can be made of the state after the motion by simply applying the prediction stage of the Kalman Filter the appropriate number of times. With this predicted state, V_S is evaluated for each of the new features. This reveals which of the features would provide the best one-off measurement after carrying out the motion blindly: it is the one which the robot will be able to measure from the most different viewpoint. This feature can therefore be chosen above the others for immediate tracking. Although the repeated measurements provided by tracking are certainly different from the one measurement at the end of the movement considered in the choice criterion, for relatively short movements at least we expect that the feature which is best to track will be the same as that best to measure once.

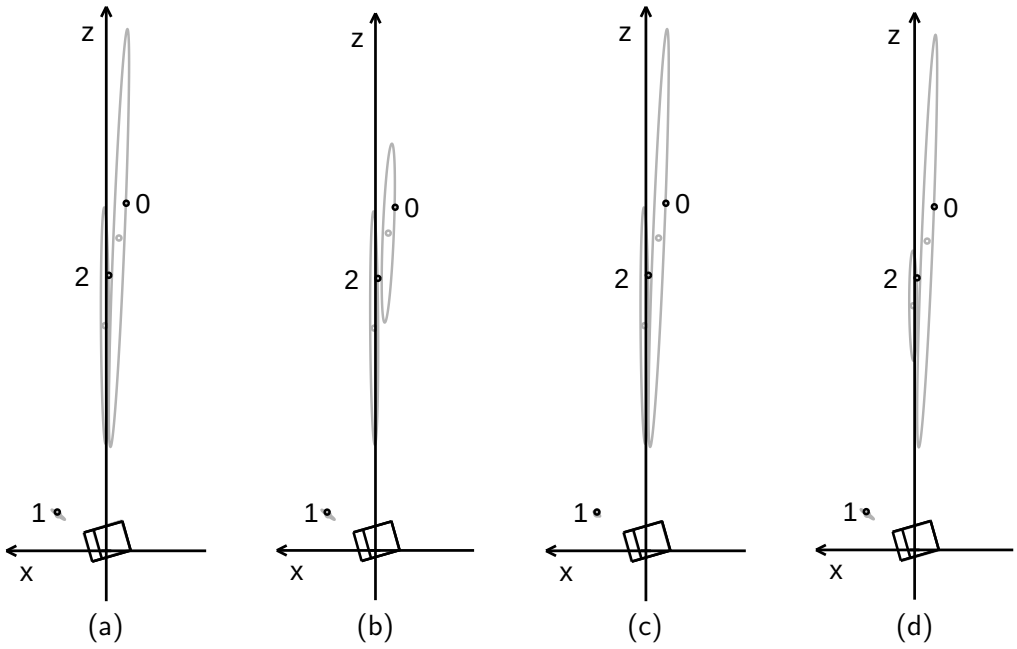


Figure 6.3: Selecting between newly-initialised features: (a) shows the state calculated from odometry only after the robot has made a tight turning motion to the left from the origin. In (b), (c) and (d) the state is shown based on tracked measurements of features 0, 1 and 2 respectively.

Experiments

To test the method for choosing between new features, from the origin the robot initialised 3 features in widely spaced positions. A motion through a tight turn to the left was planned: the state at the end of this movement was predicted, and the 3 features evaluated at this state to determine the best one for tracking. The robot then made the motion, stopping regularly to measure all the features so that estimates of its state could be made as if it had tracked each individually. Figure 6.3 shows the state at the end of the motion calculated using just odometry in (a), and after tracking features 0, 1 and 2 respectively in (b), (c) and (d). It can be seen that the odometry worked very accurately in this particular experiment: the odometry-only estimated robot state in (a) is already very close to the ground-truth.

The true robot state after the movement was:

$$\mathbf{x}_v = \begin{pmatrix} z \\ x \\ \phi \end{pmatrix} = \begin{pmatrix} 0.13 \\ 0.17 \\ 1.85 \end{pmatrix} .$$

The estimated robot state and covariance from odometry only were:

$$\hat{\mathbf{x}}_v(\text{odom}) = \begin{pmatrix} 0.13 \\ 0.18 \\ 1.87 \end{pmatrix} \quad , \quad \mathtt{P}_{xx}(\text{odom}) = \begin{bmatrix} 0.00145 & 0.00056 & -0.00224 \\ 0.00056 & 0.00168 & 0.00105 \\ -0.00224 & 0.00105 & 0.02180 \end{bmatrix} .$$

It can be seen from the covariance matrix $\mathtt{P}_{xx}(\text{odom})$ obtained for the odometry-only estimate that the largest uncertainty in the movement was in the angle ϕ through which the

robot had turned.

The estimated robot state and covariance after tracking features 0, 1 and 2 respectively were:

$$\hat{\mathbf{x}}_v(0) = \begin{pmatrix} 0.13 \\ 0.18 \\ 1.84 \end{pmatrix}, \quad \mathtt{P}_{xx}(0) = \begin{bmatrix} 0.00121 & 0.00073 & -0.00014 \\ 0.00073 & 0.00156 & -0.00025 \\ -0.00014 & -0.00025 & 0.00012 \end{bmatrix}$$

$$\hat{\mathbf{x}}_v(1) = \begin{pmatrix} 0.13 \\ 0.18 \\ 1.85 \end{pmatrix}, \quad \mathtt{P}_{xx}(1) = \begin{bmatrix} 0.00039 & 0.00011 & 0.00024 \\ 0.00011 & 0.00039 & -0.00014 \\ 0.00024 & -0.00014 & 0.00035 \end{bmatrix}$$

$$\hat{\mathbf{x}}_v(2) = \begin{pmatrix} 0.12 \\ 0.17 \\ 1.85 \end{pmatrix}, \quad \mathtt{P}_{xx}(2) = \begin{bmatrix} 0.00105 & 0.00063 & -0.00014 \\ 0.00063 & 0.00150 & -0.00031 \\ -0.00014 & -0.00031 & 0.00014 \end{bmatrix}$$

Now the values obtained in the comparison of features were:

0. $V_S = 0.00126$
1. $V_S = 0.00359$
2. $V_S = 0.00136$

A clear favourite emerged: feature 1, with 0 and 2 roughly equal but some way behind. To evaluate whether 1 really was the best feature to track, there is little evidence to be drawn from the different robot state estimates $\hat{\mathbf{x}}_v(0)$, $\hat{\mathbf{x}}_v(1)$ and $\hat{\mathbf{x}}_v(2)$: all are very close to the ground-truth in this experiment as was the odometry-only estimate. Looking at the covariance matrices $\mathtt{P}_{xx}(0)$, $\mathtt{P}_{xx}(1)$ and $\mathtt{P}_{xx}(2)$, it can be seen that measuring any of the features is good for reducing the uncertainty in $\hat{\phi}$: they all have small (3, 3) elements. This is expected: looking at any fixed point while turning will allow a good estimate of the angle to be made. We also see, though, that measuring feature 1 has done a much better job of reducing the uncertainty in $\hat{z}$ and $\hat{x}$, the estimated 2D coordinates of the vehicle on the ground plane — the (1, 1) and (2, 2) elements of $\mathtt{P}_{xx}(1)$ are much smaller than those of the other covariance matrices. The fact that feature 1 is much closer to the robot than the others has meant that the small movement which the robot has made has produced a larger change in viewpoint of this feature than the others — there has been a chance for the elevation and vergence angles to change during tracking as well as the large pan axis change experienced when tracking any of the features. The choice criterion for new features correctly identified this feature as best to track for the same reasons.

It may seem from this experiment and that of previous section that it will always be preferable to use very close features as landmarks: this is not necessarily the case. For instance, distant features have the advantage that the same ones can be measured as the robot moves through much larger distances, due to their relatively small change in appearance. In the extreme case, GPS has shown that extremely distant beacons in the form of satellites can be used for accurate localisation (although with measurements of distance, rather than distance and angle as here). The approach in this thesis treats all features in the same manner: choices will be made on grounds of how useful they are in information terms, and it will be seen in the next chapter that distant features often have a part to play.

6.3.3 Continuous Switching of Fixation

In the previous two sections, a "stop and look" approach was assumed: with the robot at a standstill, a known feature can be chosen to give the best update with an immediate measurement, or a new feature chosen to give the best update after a predicted motion. There is no significance in the position of the feature: as the robot moves off, the active head will saccade to fixation and tracking will commence. The robot will then stop after a certain time to plan its next movement and choice of feature.

Now we consider the problem of selecting between features while the robot is actually moving. Direct implementation of selection of the feature with the current highest V_S is not ideal: what happens is that once a measurement has been made of one feature, the criterion suggests an immediate saccade to another. Once this feature is found, another fixation switch is required. The result is that the active head is constantly saccading, making a quick measurement and then moving on again.

Our active head is very fast, but it takes a certain amount of time to make a saccade: for two features which are widely separated (one directly ahead of the robot and one at the side for instance), the time from starting to move until a successful measurement is made of the new feature can be more than half a second. During this time, no measurements are possible. On the other hand, at the usual tracking rate of 5Hz around 3 measurements could have been made if fixation had been maintained on the current feature. These measurements would each have been less valuable than the one finally obtained of the new feature, but it might be that together they amount to more.

To incorporate the penalty of saccading into the choice-of-feature criterion, we first require a basis for deciding whether one estimated state is better than another. Remembering that total map integrity is what is important, we suggest that in a particular state, when a comparison of the visible features is made, $V_S(\max)$, the highest V_S found, is a good indicator. If $V_S(\max)$ is high, there is a measurement which needs to be made urgently to smooth off the map's rough edges. If, however, $V_S(\max)$ is low, the relative positions of *all* the features are known well, and the state estimate is good.

The essence of our approach to choosing features while the vehicle is moving, therefore, is to predict the robot and map state at a short time in the future based on either staying tracking the current feature or a potential saccade to each alternative, and then compare $V_S(\max)$ for these states to see which is lowest. The "short time in the future" is chosen to be the length of time in which a saccade and measurement could be successfully completed to any of the features. The steps followed are:

1. Calculate the number of measurements N_i which would be lost during a saccade to each of the visible features. This is done by estimating the time which each head axis would need to move to the correct position, taking the largest (usually the pan time since this axis is the slowest), and dividing by the inter-measurement time interval (0.2s).

2. Identify $N_{\max}$, the highest N_i: this is the number of measurements lost in the largest saccade available.

3. For each feature i, make an estimate of the state after $N_{\max}+1$ steps if an immediate saccade to it is initiated. This consists of making N_i filter prediction steps followed

by $N_{\max} - N_i + 1$ simulated prediction/measurement updates. A measurement is simulated by updating the state as if the feature had been found in exactly the predicted position (it is the change in covariance which is important here rather than the actual estimate). An estimated state after the same number of steps is also calculated for continued tracking of the currently selected feature.

4. For each of these estimated states, $V_S(\max)$ is evaluated. The saccade providing the lowest $V_S(\max)$ is chosen for action; or tracking stays with the current feature if that $V_S(\max)$is lowest.

One important point here is that the efficient updating method of Section 6.2 cannot be used in this continuous fixation switching framework: we rely on the full state and covariance being up-to-date at all times.

Experiment with Fixation Switching Between Four Features

Figure 6.4 shows an experiment into continuous fixation switching: four features were initialised, and (a) shows the robot's trajectory as it started to move forward at about 4cms^{-1}, choosing which features to fixate on as described above. In (b), the values obtained from a straight V_S comparison of the four features at each time step are plotted. The four lines show how uncertainty in the positions of the features relative to the robot vary with time. As would be hoped, there is a general downward trend (from the initial state where all the features have $V_S = V_{S(\text{new})}$ as explained in Section 6.3.2), showing that the positions are becoming more and more certain.

In the early stages of the motion, fixation switches as rapidly as possible between the four features: only one measurement at a time is made of each feature before attention is shifted to another. In the graph of Figure 6.4(b), a measurement of a particular feature appears as a sharp drop in its V_S value. While a feature is being neglected, its V_S gradually creeps up again. This is because the newly-initialised features have large and uncoupled uncertainties: their relative locations are not well known, and measuring one does not do much to improve the estimate of another's position. After a while, the feature states become more coupled: around step 40, clear zig-zags in the rising curves of neglected features show that the uncertainties in their positions relative to the vehicle are slightly reduced when a measurement is made of *another* feature.

At around step 80, the first clear situation is seen where it becomes preferable to fixate one feature for an extended period: feature 1 is tracked for about 10 steps. This feature is very close to the robot, and the robot is moving towards it: measurements of it provide the best information on the robot's motion (the motion past feature 0 is rather tangential and prone to the ambiguity of Figure 5.7). Since the locations of the other features are becoming better known, their positions relative to the robot are constrained quite well by these repeated measurements (only a gentle rise in the lines for features 0, 2 and 3 is seen during this time). Feature 1 actually goes out of the robot's view at step 101 (the robot having moved too close to it, violating one of the visibility criteria of Section 4.1.2), and behaviour returns to quite rapid switching between the other features.

The robot was stopped at the end of this run with state estimates intact. It was then driven back to near the origin in a step-by-step fashion, making further dense measurements of all of the features along the way. The result was that once it was back at its starting

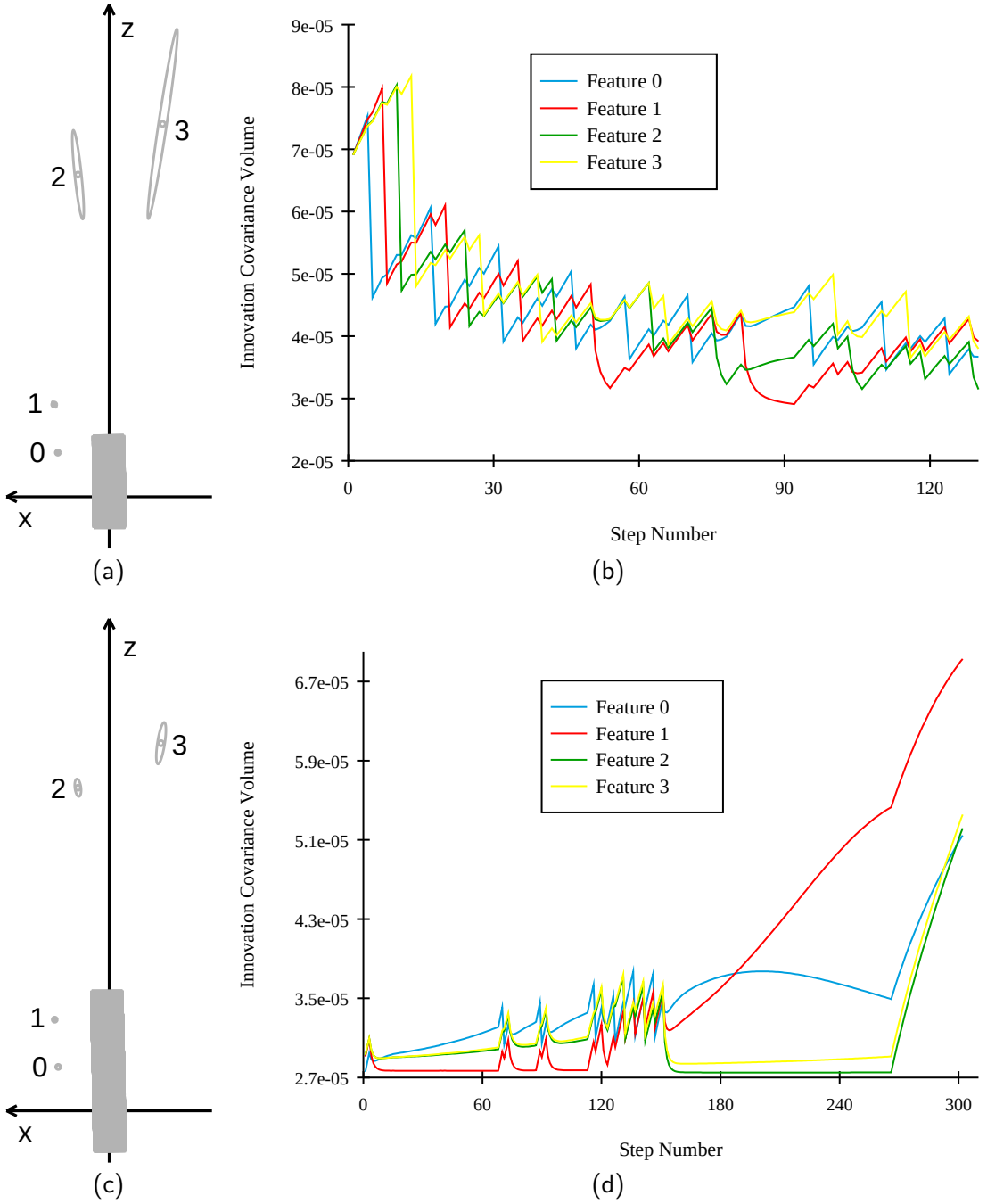


Figure 6.4: Experiments in continuous fixation switching: for the trajectory in (a), as the robot moves in the region of 4 newly-initialised features, (b) compares the innovation covariance volume V_S for the features as fixation switches between them. In a second motion in (c), the world positions of the features are more accurately known, and (d) again compares V_S.

point, feature estimates had been very well established. It was from this point that a second continuous switching run was initiated: the trajectory and the now accurately estimated feature positions are shown in Figure 6.4(c), and a graph of the feature comparison in (d).

This second graph is dramatically different from the first: in the early stages, low V_S values for all the features are now maintained by extended periods of tracking one feature (feature 1 again). The strong coupling now established between feature estimates means that if the robot position relative to one can be well estimated, as is the case when the nicely placed feature 1 is tracked, its position relative to the others will be as well. There is the occasional jump to another feature, appearing as spikes in the traces at around steps 70 and 90. Just after step 120, feature 1 goes out of view, and a period of rapid switching occurs. None of the remaining features on its own provides especially good overall robot position information, and it is necessary to measure them in turn.

Feature 0 goes out of view (due to too large a change in viewing angle) at step 147. After this, only the distant features 2 and 3 remain for measurements. It is noticeable that throughout the graph these two have been locked together in their V_S values: measurements of them provide very similar information due to their proximity, and there is little need to switch attention between them. These features finally go out of view at about step 270, leaving the robot to navigate with odometry only.

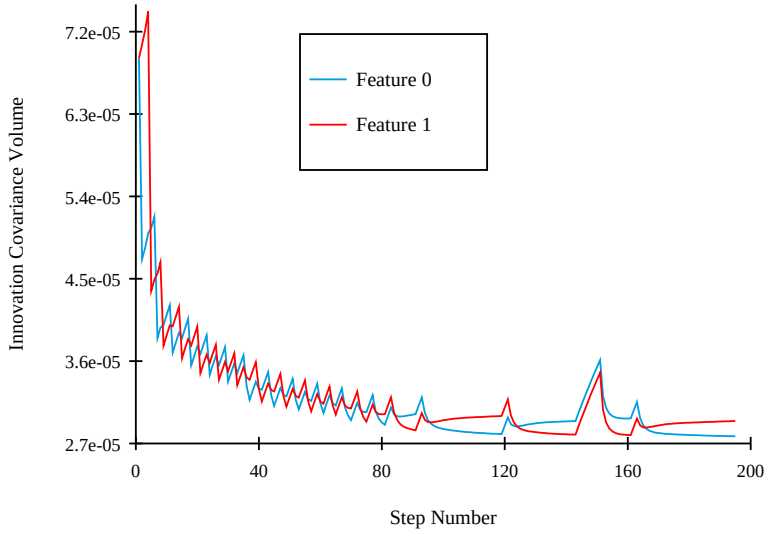
Experiment with Varying Head Performance

Our active head is very fast, but it was interesting to see how the continuous fixation switching criterion would direct the actions of a head with lower performance. To simulate this, the control parameters of Yorick 8-11 were tuned down to reduce its performance by about 30%. This change was reflected in the fixation switching method, where estimates of the time taken to make saccades were increased by appropriate amounts.

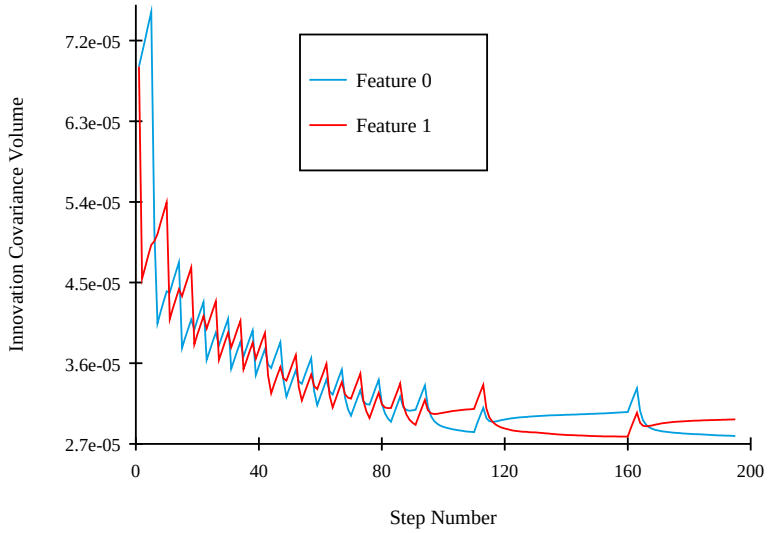
An experiment was performed with the head in both its normal configuration and in the tuned-down version. Two distant features (the same ones which were features 2 and 3 in the previous experiment) were initialised from the origin and the robot drove straight forward, switching attention between them. Graphs from the two runs can be seen in Figure 6.5, where (a) is at full-performance and (b) tuned-down. The differences are slight but clear: the traces for the two features are bound more tightly together in (a) in the early stages where repeated switching occurs: the more rapid saccades available allow the robot to keep its uncertainty in the two features close. Later on, when the feature estimates have been coupled, we see fewer fixation switches in (b), where the larger penalty of saccades means that it is worthwhile tracking one feature for longer. Note that the particularly large spike in graph (a) at around step 150 is caused by some failed measurements.

6.4 Conclusion

We have shown that features can be tracked continuously in an efficient manner as the robot moves, and that intelligent choices can be made about which features should be tracked when to provide good navigation information. In the next chapter, these methods will be integrated into a system for automatic map-maintenance and localisation for use in goal-directed navigation.



(a)



(b)

Figure 6.5: Varying the performance of the active head to see the effect on continuous fixation switching between two features. In (a), the head is at normal full performance, whereas in (b) it has been reduced by about 30%.

One question arises concerning the choice-of-feature criteria derived in Section 6.3: the assumption has been made that the best measurement to make is that which can in one step reduce map uncertainty the most — and this will be a measurement whose result is uncertain. In Section 6.3.3, the factor of potential measurements being weighted against by the time necessary to saccade to a new feature was taken into account, but what has not been considered is the fact that the most uncertain measurements will take the most computational time to acquire, in terms of large image regions which will have to be searched. In a system such as ours with mechatronic active vision, this factor is perhaps insignificant compared with the effort of changing fixation to another feature. However, with more general active sensors, such as a digitally movable window within an image from a static camera, is there an optimum size of search window to investigate, or is it just as useful to make many fast, predictable measurements rather than a few unpredictable ones?

7

Autonomous Navigation

In this final major chapter, we will look at how the methods developed in this thesis will form the key parts of a useful autonomous robot navigation system using active vision. As discussed in Chapter 1, it must be remembered that navigation in unknown surroundings is a challenging goal, and one which has not yet been achieved satisfactorily by mobile robots operating without external assistance. Robots have been built which can move autonomously for long periods in unfamiliar areas, such as the lawn-mowing robot mentioned in Chapter 1, but not while monitoring their location relative to any fixed axes, which we argue is necessary for many tasks. Those robots which have shown success in navigating with respect to fixed axes have usually made use of prior knowledge about the environment, such as beacons in known positions.

If we assume that the robot's overall goal is decided by some other process — perhaps a pre-programmed position command from a human operator, a target-seeking behaviour pulling the robot towards some stationary or moving object, a desire to move in a looped patrolling motion, or an exploration behaviour pulling it towards new areas — what capabilities does it need to accomplish that goal? It must be able to:

- Monitor its position as it moves.
- Plan and control its movement.
- Avoid obstacles.

The first of these is what the main part of this thesis is about: by building and using a map of landmark features, the robot is able to obtain much better localisation information than it would otherwise obtain from solely odometry. We will look in Section 7.1 at how a useful map is automatically generated and maintained in our system.

The second capability, planning and controlling motion, is perhaps relatively simple in relation to the other two, particularly in the case of our robot which is assumed to move on a perfect planar surface. In Section 7.2, a simple method is described via which the robot may either steer towards a waypoint specified by ground-plane coordinates, or steer safely around a known obstacle. It is shown how these types of motion can be combined sequentially to produce compound movements.

In Section 7.3 we present an experiment which demonstrates automatic map-maintenance and motion control as the robot moves autonomously up and down a corridor-like area, navigating to specified waypoints and matching features after periods of neglect to recognise previously visited areas. A comparison is made with a repeat of the experiment where the robot did not use visual map-building and relied unsuccessfully on its odometry.

Section 7.4 concludes the chapter with discussion and several experiments indicating what additional capabilities it is felt are needed to take autonomous robots into the real world, with vision as the primary sensing modality. Position-based navigation is contrasted with a context-based methodology, and the route to their fusion in truly capable robots is discussed.

7.1 Maintaining a Map

We have seen from the experiments of the previous two chapters that robot localisation estimation is always improved by tracking fixed landmarks in the world. Further, the best scenario is to make repeated measurements of the same set of landmarks as the robot moves within a particular area, and come back to previously seen landmarks when an area is revisited. This is possible if a map of features is generated and maintained in which each can be relied upon to be matchable and have a well-known location. The Kalman Filter of Chapter 5 ensures that the best use will be made of whatever measurements are obtained to build a consistent map, and it was shown how features can be added or deleted at will; in this section we will consider how choices should be made about which features to include in the map.

As mentioned in Section 6.3.2, the likelihood is that there will only be a limited number of features in a particular environment which will make good landmarks, in terms of having sufficient contrast with the background and viewpoint-invariant appearance to be matched reliably. For this reason, initialisation of new features is performed somewhat arbitrarily: no attempt is made to search for features in particular positions. The active head is directed to view the scene to the left, right and straight ahead of the robot, and at each of these positions the patch-finding operator of Section 4.1.1 is applied in the images obtained and an attempt is made to initialise a feature if a good patch is found. In this way, up to 3 widely-spaced new features will be added to the map. A choice of which of these to track first is then made as in Section 6.3.2.

During general autonomous navigation, choice of the feature to track at a given time is made according to the V_S criterion presented in Section 6.3.1: each feature in the map is tested for its expected "visibility", meaning whether the robot is estimated to be within an area from which it will have a view of the feature which is sufficiently close in distance and angle to the viewpoint from which the feature was initialised (see Section 4.1.2). The visible feature which will provide the most information is then selected for measurement.

When is it time to initialise more new features? Obviously at the start of a journey, when no landmarks are known. After this, the map-building formulation allows more to be added at any time. In some ways, it is preferable for the map not to be too dense: having a large number of landmarks means that fewer measurements can be made of each one, and their positions will be less well known. The automatic feature selection criterion will choose the newer, less well-known features, and while these are being tracked the well-known ones can go out of view, meaning that the robot position estimate will tend to drift. Also, it is preferable in terms of efficiency to keep the complexity of the map down.

Remembering the importance of repeated switching of tracking demonstrated in the experiments of the previous chapter, but also aiming to keep the map relatively sparse, it was decided that at least two features should be available for measurements at a given time. Therefore, the robot stops and initialises new features if, due to motion into a new area or the deletion of features, a point is reached where less than two are visible.

It was mentioned in Chapter 4 that when a feature is initialised no attempt is made to determine whether it will make a durable landmark. This is not strictly the case, because the repeated stereo matching between the robot's two cameras necessary to fixate a feature during initialisation ensures that it at least has viewpoint-invariance across the relatively small inter-ocular baseline. However, newly-initialised features will in general not have been confirmed as good landmarks. The natural approach which can be taken to decide whether they are is to see how successfully they can be tracked. Making repeated measurements of a feature under the constraints of the Kalman Filter means that by definition it makes a good landmark: it must be a fixed point in 3D space which does not change greatly in appearance with viewpoint. It is possible to conceive of features which could deceive the filter by remaining trackable while changing their position: an apparent corner feature at an occlusion boundary, for instance, might move in 3D by a small enough amount with viewpoint changes to convince the robot that the feature was actually stationary and the robot's motion was slightly different than would otherwise be calculated. Fortunately, these are rare in practice, and anyway with frequent enough measurements of a range of other features the robot's motion can be checked against other references.

To decide which features to keep in the map, therefore, for each one counts are maintained of the number of attempted measurements and the number of successful measurements. Since measurements are only attempted from robot positions form which the feature is expected to be "visible", the ratio of these gives a score reflecting how reliable the feature is. A feature is deleted from the map if, after at least 10 attempted measurements, the successful measurement ratio ever falls below 0.5. Deletion will occur if the feature is frequently occluded as well is if it is a "bad" feature as discussed above.

The criterion for deleting features from the map works well in practice. If a feature has been successfully measured a lot of times from within its area of expected visibility, it builds up a high scores for both attempted measurements and successful measurements: the ratio of these is then quite stable. A few missed measurements of the feature do not lead to its deletion, and rightly so: the reason for these missed measurements is something other than the feature's quality. It is possible that the robot has temporarily badly estimated its own location and assigned too much certainty to this estimate. The failed measurements confirm this, and the robot's position uncertainty will grow (in the standard prediction stage of the filter), hopefully to the point where the feature, still safely in the map, will lie within the larger search regions generated.

In summary, the approach taken to automatic map maintenance is:

- From the visible features, choose the current one to track according to the information criteria of Section 6.3.
- If fewer than two features are currently visible, initialise some new ones.
- If more than half of the attempted measurements of a feature have failed (after at least 10 attempts), delete it.

The results in practice will be seen in Section 7.3.

7.2 Controlling the Robot's Motion

With the information from its localisation and map-building system, the robot is in a position to control its movements either in relation to the world coordinate frame or in relation to features in the map. A similar approach is taken in both cases, which makes use of the ideas of visual servoing [44]: if an object to be interacted with is tracked, the small changes in the measurement of its position at each step can be directly coupled to the control system of the robot attempting the interaction. The absolute locations of either manipulator or objects need not be known.

If a feature can be tracked at fixation by an active camera or a stereo head, servoing information is directly available in a very useful angular form. The values from the joint angles provide proprioceptive information which abstracts and replaces the original image measurements. The next section shows how the robot can steer around an obstacle feature which is tracked at fixation using our normal filter approach (Section 7.4 will consider how to incorporate this behaviour into autonomous navigation), and how a similar approach can also be used to steer the robot towards specified waypoints on a desired trajectory.

7.2.1 Steering Control

A simple law is used to control the robot's steering angle when navigating either around an obstacle [6] or towards a journey waypoint.

Steering Around An Obstacle

Referring to Figure 7.1, an obstacle exists at O which the robot must avoid by a safe radius R. From a fixated stereo measurement, the pan angle α and distance D to the feature are recovered. It can be seen that if the robot were to turn on the spot about the head centre through the angle $\alpha - \sin^{-1}\frac{R}{D}$, it would be directly lined up to drive along a tangent to the safe-radius circle. Rather than do this, the angle s of the steering wheel at the back of the vehicle is set to be proportional to this required correction in heading:

$$s = \kappa(\alpha - \sin^{-1}\frac{R}{D}) \ . \tag{7.1}$$

κ is a constant of proportionality called the steering gain. As the robot starts to drive towards the obstacle steering according to this law, repeated measurements are obtained from the active head, and the steering angle continuously changes. The result is a smooth

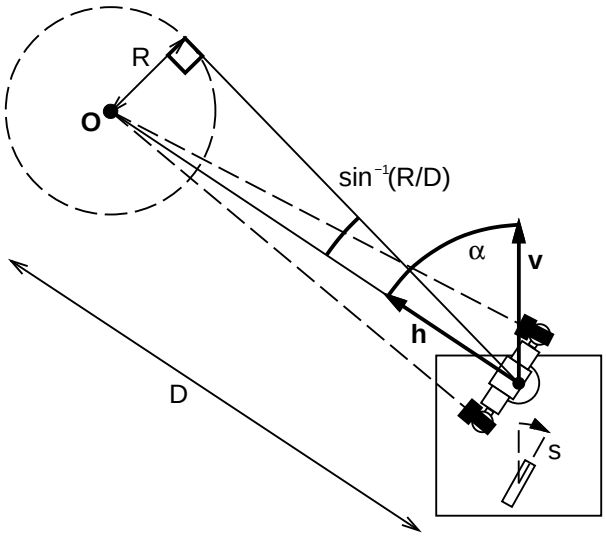


Figure 7.1: Steering at a safe radius R around an obstacle at O: *alpha* is the pan or gaze angle to fixate on the feature, and D is its distance from the head centre. s is the steering angle derived.

steering trajectory directing the robot around the obstacle. The value of κ affects how urgently the robot will turn towards the tangent circle — low values produce more gentle trajectories. Figure 7.2 shows the action of this steering law in real experiments where the robot steered around an obstacle with different initial relative obstacle/robot positions and different values of gain κ. The obstacle was a clearly visible feature on the corner of a laboratory bench (the top four images of Figure 6.1 were taken from one of these experiments), chosen because it was matchable over the long robot motions involved.

Steering Towards a Waypoint

With a small adaption, the steering law can be used to direct the vehicle smoothly towards a specified waypoint rather than around an obstacle, in a form of control relying on the robot's world position estimation. The ground-plane coordinates (Z_w, X_w) of the point through which it is desired that the robot should pass are used to calculate the pan angle α_w to which the active head would have to be turned to observe a feature at that position according to the current robot location estimate. The steering angle is then set proportional to this, but without the safe radius offset of Equation 7.1:

$$s = \kappa_w \alpha_w \ . \tag{7.2}$$

This law will tend to steer the robot directly towards the waypoint, the value of κ_w again playing the role of a gain parameter affecting the urgency of steering.

A long planned trajectory consists of a series of waypoints to be reached in sequence. A waypoint is said to have been reached if the robot's estimated location lies within a small radius of the waypoint position, and attention is now turned to the next one. If the pan angle to a waypoint is ever so large as to make the steering angle s calculated from

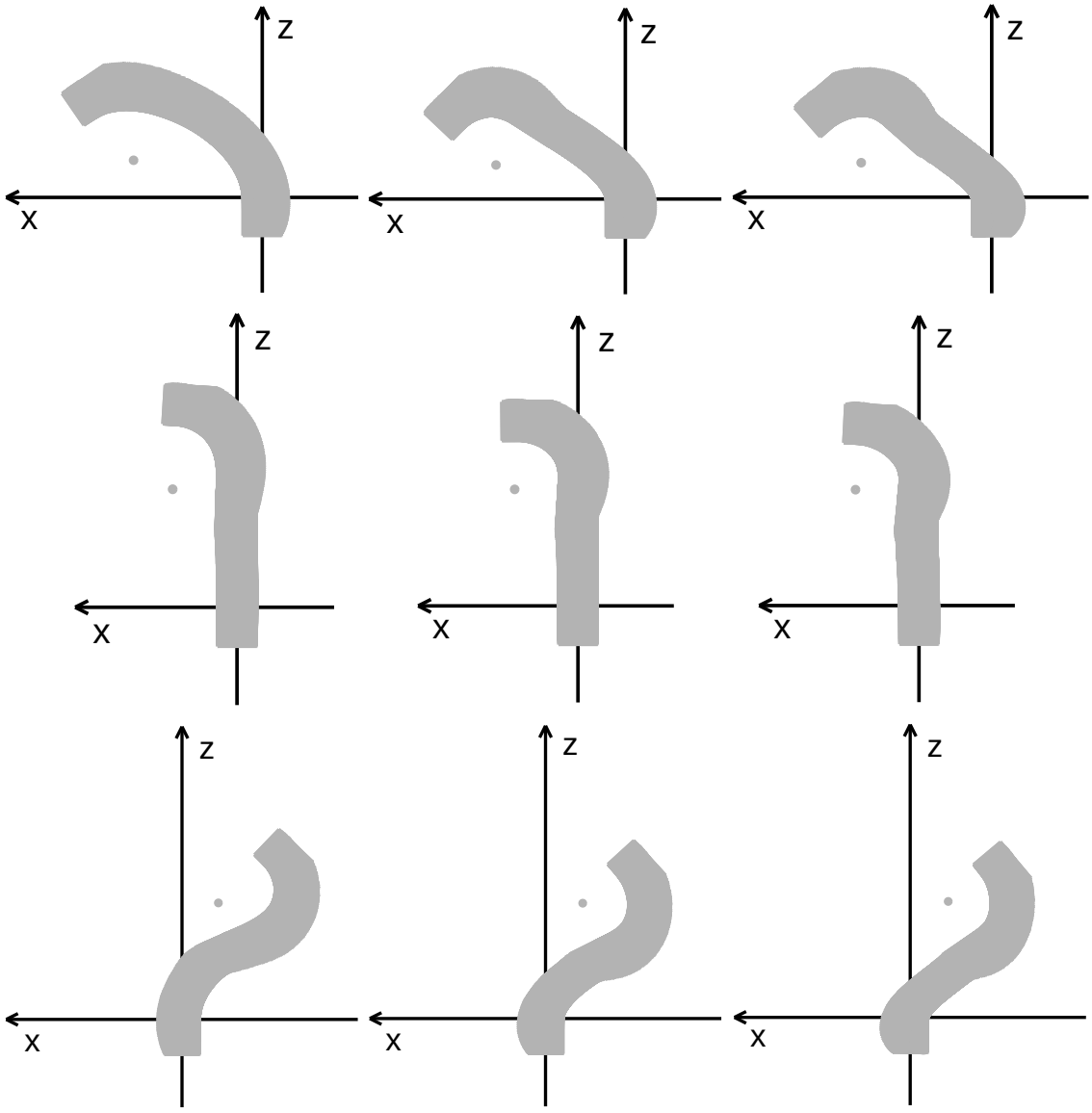


Figure 7.2: Robot trajectories produced by the the steering law with different values of gain κ and different obstacle positions.

Equation 7.2 greater than $\pi/2$, s is limited to this value: the robot makes a turn on the spot. This is most likely to occur when a new waypoint has just become the steering goal.

A note on velocity control: during map-building, the robot travels at a velocity fixed at the start of the run, but it moves more slowly when sharp turns are called for. This makes tracking features slightly easier for the filter since their angular movement between measurement steps, and therefore the area of image search regions, are reduced. Of course, the choice of velocity does not affect the trajectory of the robot.

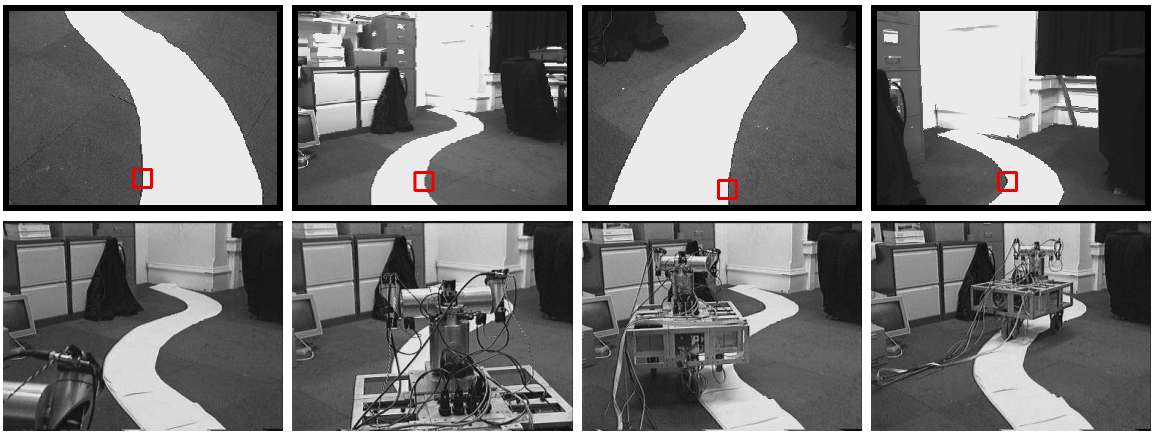


Figure 7.3: Road-following using fixation on the "tangent point" of the inside kerb: the top four pictures show the robot's view as it fixates the road edge and the bottom four its progress in frames cut from an externally taken video.

7.2.2 Road-Following by Visual Servoing

As a temporary aside from the map-building approach of the rest of this thesis, the obstacle-servoing method of the previous section can be extended to the problem of directing a robot along a winding road or track [68]. Fixation is not now on a fixed point, but on the continuously sliding "tangent point" of the inside kerb of the road. This is the point where the line-of-sight of the robot meets the road's edge at a tangent, and it is easily detected in images as the point where the road's curving edge is vertical. The robot's steering angle is then set according to Equation 7.1. The safe radius R now has the role of a desired safe distance from the kerbside.

If more than one tangent point is visible, the most distant to which there is a line-of-sight unobstructed by closer kerbs is selected. A saccade is initiated when the point selected by this criterion crosses to the other side of the road. In this way, the robot can navigate a sequence of corners.

Figure 7.3 shows road-following in practice along a winding track laid out in the laboratory, with views from the robot's camera fixated on the current tangent point and frames cut from a video of the journey. Note that in this experiment the active head was controlled so as to fixate the point of interest near the bottom of images: this was to provide the largest possible look-ahead in the upper image portion so that the next fixation point could be located reliably while fixation was maintained.

This method is interesting because remarkable parallels can be drawn with the way that human car drivers observe the road and steer their vehicles. Land and Lee [49] and Land and Horwood [48] conducted experiments in which drivers' eyes were tracked to estimate their fixation points on the winding road ahead. They found that for a large proportion of the time attention was concentrated on the inside kerb of the current bend. Also, horizontal viewing angle and the angle to which the steering wheel was currently being turned showed a large degree of correlation. It appeared that steering inputs were approximately proportional to gaze angle, as is the case in our robot implementation (since the term $\sin^{-1}\frac{R}{D}$ in Equation 7.1

becomes negligibly small when the distance D to the fixation point is large compared with the safe distance R, which is true in real driving).

7.3 Automatic Position-Based Navigation

Returning to our main theme, using automatic feature-selection, map maintenance and steering capabilities, the robot is in a position to perform autonomous position-based navigation. A trajectory is specified as a sequence of waypoints in the world coordinate frame through which the robot is desired to pass. The robot moves in steps of approximately two seconds duration. Before each step, feature selection chooses the best landmark to track during the movement, and features are added to or deleted from the map if necessary. As the robot drives, making measurements of the chosen feature and updating the localisation filter at 5Hz, the steering angle is continuously set to the appropriate value to reach the next waypoint.

In an experiment, the instructions given to the robot were to head in sequence from its starting point at $(z, x) = (0, 0)$ to the waypoints $(6, 0.4)$, $(6, 0)$, and finally $(0, 0)$ again (in metre units). This experiment was designed to prove again the system's ability to return to a previously visited area and recognise it as such, but now using a map which was generated and maintained completely automatically. Note that the extra waypoint $(6, 0.4)$ was specified simply to ensure that the robot turned in a way which was predictable and did not tangle its umbilical cable.

7.3.1 Using Odometry Only

To first demonstrate the real need for vision or some other kind of sensing in a robot motion of this kind, the experiment was run with no map-building and the robot relying on odometry only for its position estimation. This run was the same in other respects to the full experiment below (the robot moved in the same kind of steps and controlled its motion in the same way).

The state at several numbered steps is shown in Figure 7.4. There is quite a large amount of drift between the ground-truth and estimated robot positions from early on in the motion, and with no feature measurements this is never rectified. By step (64), the experiment had to be stopped because although the estimated state was that the robot was heading right down the centre of the corridor, it was actually veering off to the side and was about to crash. The (z, x, ϕ) true and estimated robot positions, and the covariance of the estimate, were:

$$\mathbf{x}_v = \begin{pmatrix} 2.89 \\ -0.29 \\ -2.85 \end{pmatrix} \quad , \quad \hat{\mathbf{x}}_v = \begin{pmatrix} 2.72 \\ 0.05 \\ -3.12 \end{pmatrix} \quad , \quad \mathtt{P}_{xx} = \begin{bmatrix} 0.0018 & -0.0010 & 0.0005 \\ -0.0010 & 0.0151 & -0.0039 \\ 0.0005 & -0.0039 & 0.0038 \end{bmatrix} .$$

The estimation is particularly bad with respect to the robot's orientation ϕ, where a discrepancy of 0.27rad or around 15° is seen. Of course, the system knows that its position estimate is unreliable, and the estimated position covariance is large (although not quite as large as it should be in this case to account for the discrepancy, showing potential undersetting of noise in the vehicle motion model, or more likely that the motion noise model is

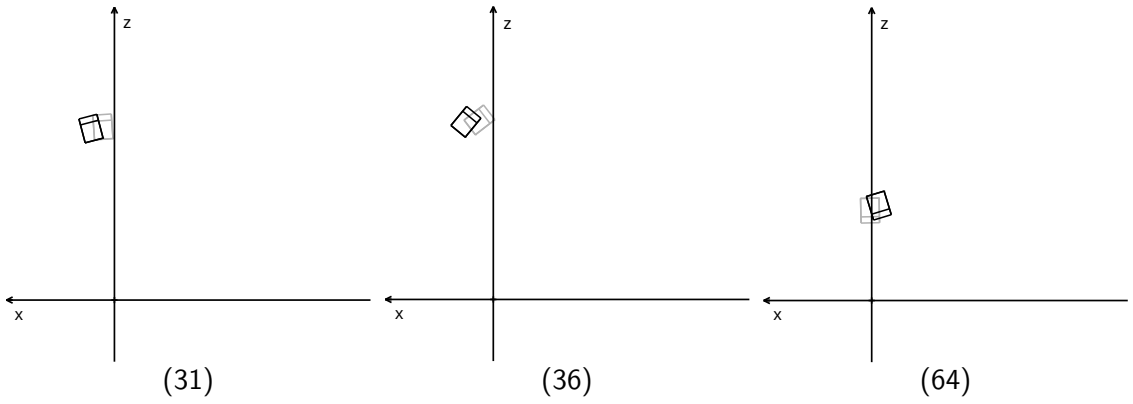


Figure 7.4: Position-based navigation using odometry only: the robot is controlled with respect to its estimated position (shown in grey), so this follows the desired route between waypoints. However, the estimated robot position is continuously drifting from the ground-truth (in black).

not always adequate to account for the systematic error sources mentioned in Section 3.4.1). A more advanced motion planner could take account of this when for example trying to avoid an obstacle in a known location: it would be potentially unsafe to move into any region where the robot's uncertainty spread overlapped the obstacle.

7.3.2 Using Vision

Now employing its map-building system, the robot's progress is shown in frames cut from a video in Figure 7.5, along with views output by the system of the first 15 features detected and initialised into the map. The output of the filter at various numbered steps appears in Figure 7.6, where those features are annotated. Some features did not survive very long before being abandoned as not useful. (Numbers 4 and 5 in particular did not survive past very early measurement attempts and do not appear in Figure 7.6 — 5 in particular was clearly never going to make a good fixed landmark, being an image on a monitor attached to one of the robot's cameras!) Others, such as 0, 12 and 14 proved to be very durable, being easy to see and match from all positions from which they are expected to be visible. It can be seen that many of the best features found lie near the ends of the corridor, particularly the large number found near the cluttered back wall (11–15, etc.). The active approach really comes into its own during sharp turns such as that being carried out at around step (44), where features such as these could be tracked continuously, using the full range of movement of the pan axis, while the robot made a turn of 180° . The angle of turn can be estimated accurately at a time when odometry data is unreliable.

Outward Journey: the sequence of features selected to be tracked in the early stages of the run (up to step (21)) was 0, 2, 1, 0, 2, 1, 3, 5, 4, 7, 6, 8, 3, 6, 8, 7, 3, 7, 8, 3, 9 — we see frequent switching between a certain set of features until some go out of visibility and it is necessary to find new ones.

Return to Origin: at step (77), the robot had reached its goal, the final waypoint being a return to its starting point. The robot had successfully re-found original features on its return journey, in particular feature 0 whose position was very well known, and this

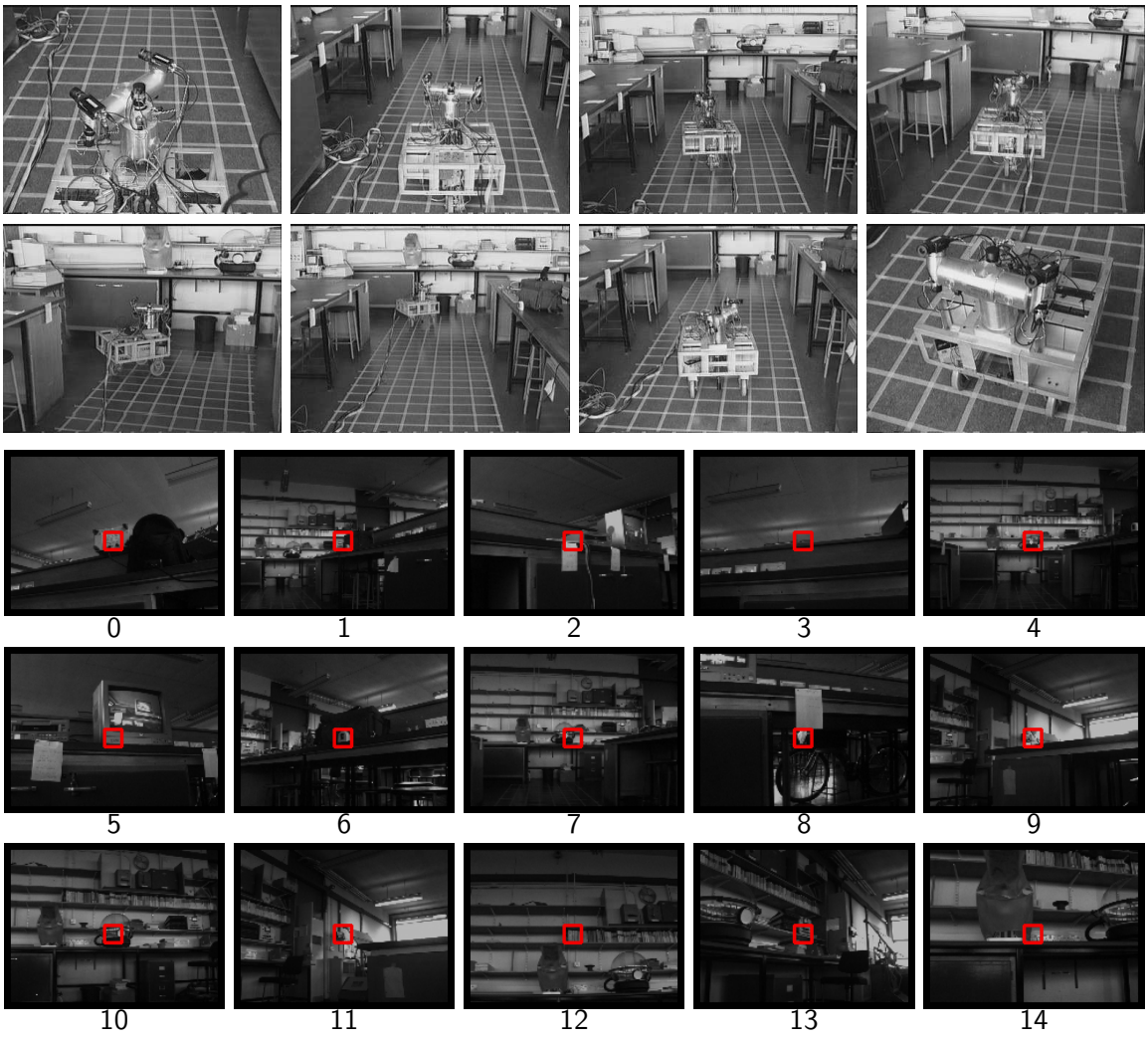


Figure 7.5: Frames from a video of the robot navigating autonomously up and down the corridor where the active head can be seen fixating on various features, and fixated views from one of its cameras of the first 15 features initialised as landmarks. The time taken for the return journey was about 8 minutes.

meant its position estimate was good. What is very appealing is that the choice of feature criterion described in Section 6.3 had demanded re-measurement of these early features as soon as they became visible again, due to the drift which had occurred between the robot position estimate and the world coordinate frame. Generally, the uncertainty in the relative positions of features and the robot, and therefore the V_S score indicating the value of measuring them, will increase with the amount of distance travelled since they were last measured. This gives the system a natural tendency to re-register itself with features seen earlier whenever possible and create a tightly known and globally consistent map.

The robot's true position relative to the gridded floor was measured here, and can be

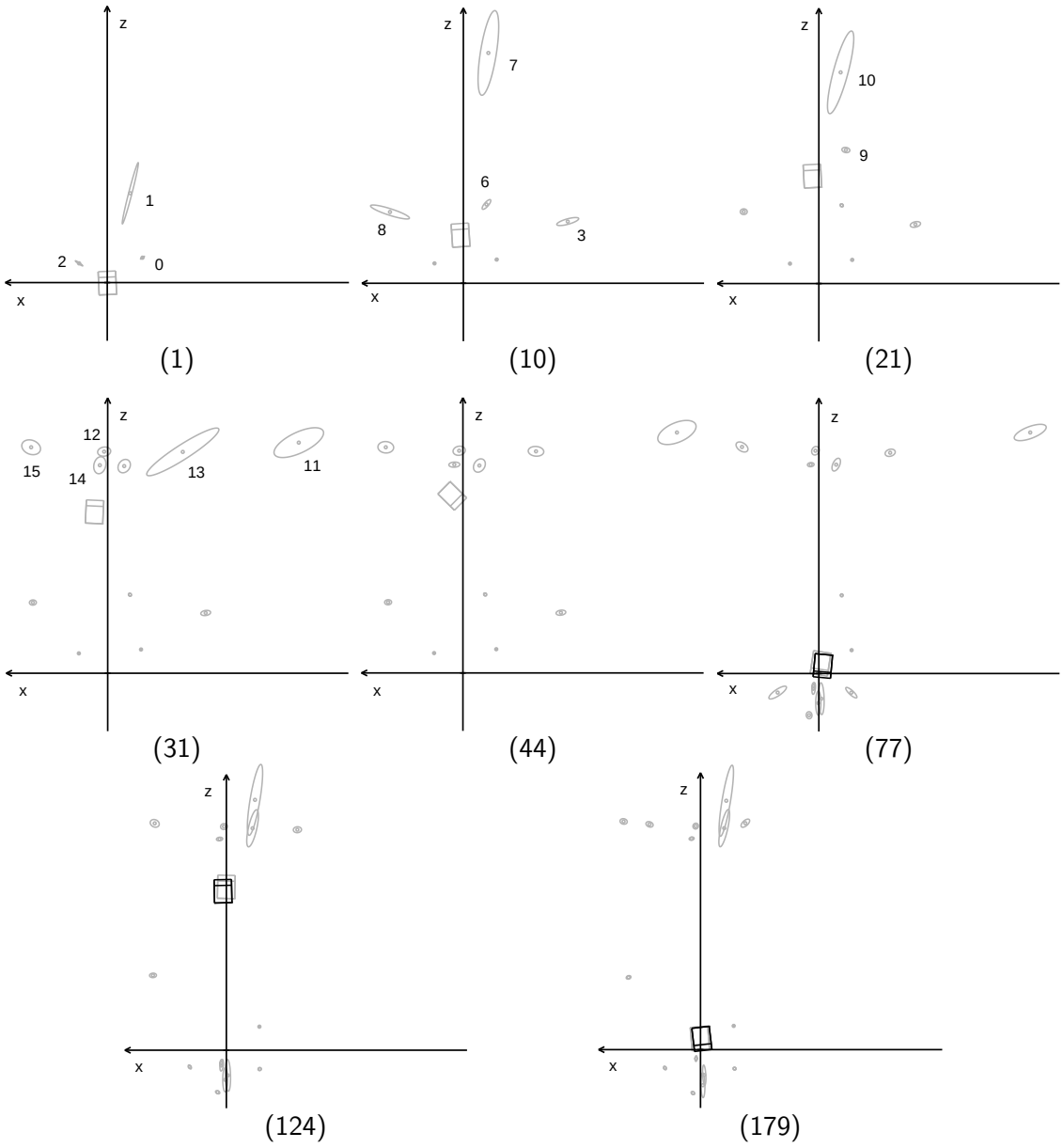


Figure 7.6: Numbered steps in autonomous navigation up and down a corridor. Grey shows the estimated locations of the vehicle and features, and black (where measured) the true vehicle position. The furthest features lie at $z \approx 8$m.

compared with the estimated state and its covariance:

$$\mathbf{x}_v = \begin{pmatrix} 0.06 \\ -0.12 \\ 3.05 \end{pmatrix} \quad , \quad \hat{\mathbf{x}}_v = \begin{pmatrix} 0.15 \\ -0.03 \\ 2.99 \end{pmatrix} \quad , \quad \mathtt{P}_{xx} = \begin{bmatrix} 0.0003 & 0.0001 & -0.0001 \\ 0.0001 & 0.0001 & -0.0001 \\ -0.0001 & -0.0001 & 0.0002 \end{bmatrix}$$

The estimate is much better than at the same point in the odometry-only experiment, although the covariance matrix is again a little too small, showing slight over-confidence in

the estimate.

Repeat Journey: to underline the usefulness of the map generated, the experiment was continued by commanding the robot back out to $(z,\ x) = (6,\ 0)$, then home again to (0, 0). In these further runs, the system needed to do far less map-building since a large number of features was already known about along the trajectory. It can be seen that many of the features previously found survive right until the end of this final run, although the fallibility of some is found out with the further measurements and they are deleted. At (6, 0), step (124), the robot's true position was $z = 5.68$m, $x = 0.12$m, $\phi = 0.02$rad, and estimated state was $z = 5.83$m, $x = 0.00$m, $\phi = -0.02$rad. At (0, 0) again, step (179), the true position of $z = 0.17$m, $x = -0.07$m, $\phi = -3.03$rad compared with the estimate $z = 0.18$m, $x = 0.00$m, $\phi = -3.06$rad. This is very impressive localisation performance considering that the robot had travelled a total distance of around 24m by this stage.

7.4 Fully Autonomous Navigation

Simply controlling the robot with respect to position waypoints specified in the world co-ordinate frame, as in the previous experiment, is useful but quite limited, even when the robot is required just to move along a pre-ordained path. The problems are:

- Once the robot is some distance from its starting point, its position estimate relative to the world frame will be unreliable, and so manoeuvring such that the estimated position passes through world waypoints will not be accurate.
- While it is using visual landmarks for localisation, the robot is effectively still moving blindly in position-based navigation. No attempt is made to check that the areas to be passed through are safe and free of obstacles, or that particular regions of interest have been reached.

Consider an example application: a robot is required to make autonomous inspection tours of a factory along a pre-ordained route. This could be implemented by specifying waypoints relative to a ground-plan of the factory, and commanding the robot to visit them in sequence, performing map-building as usual. The waypoints might lie at the centres of intersections in the factory's corridor system.

The robot would set off and happily negotiate the corridors near to its origin where it is able to measure its position accurately — just as in Section 7.3. We note from the experiment in that section, however, that when the robot had reached its furthest point from the origin, a distance of only about 6m, its z and x position estimates were both around 15cm from the ground-truth. This is not due to an inadequacy of the filter, which is doing the best job possible with the information is has, but rather a consequence of the fact that while only using automatically-acquired landmarks many measurements have to be compounded to produce a position estimate out here, as discussed in Section 5.2.1. When the robot travelled farther than this, its estimate would become dangerously uncertain (if its transverse position estimate was incorrect by around a metre, for example, the robot would be in danger of crashing into the sides of most corridors).

In Section 7.4.1, we will describe one way in which position-based control can be improved in certain environments by providing a small amount of prior information in the form of a few known features, allowing the robot to estimate its position relative to the world

frame much better over a wide area. However, this is certainly not the generally-applicable solution which will allow autonomous navigation in many situations. Even when the environment is well-known beforehand, this method does not allow for changes to be dealt with, such as the introduction of an obstacle. Further, it requires a very accurate description of the relative locations of the known features introduced, which will not be available in most situations. If the world positions of the known features are inaccurate, the robot will not be able to find them, or worse be misled by measurement of them.

A more sensible approach is to work towards supporting the same kind of context-based rather than position-based navigation which humans use in known areas. As discussed in Chapter 5, it does not seem that humans have, or necessarily need, much idea of the global relationships between navigational landmarks, and the example of moving around a well known city was given — streets are located and followed one-by-one, with only the relationships between neighbours needing to be well known. Some success has already been reported in this type of robot navigation. At the company Applied AI Systems in Canada, a Non-Cartesian Robotics approach based on the subsumption architecture of Brooks [14] has used topological maps and neural-network based visual landmark recognition to guide autonomous robots around simple paths. Their robots do not have an estimation of their overall location in a world frame, but know how to move from one landmark to the next. We will look in Section 7.4.3 at whether it is possible or sensible to integrate this methodology with our map-building and localisation system.

The problem of classifying free space/obstacle regions and the ways in which this could become part of an overall navigation system using active vision is investigated in Section 7.4.2.

7.4.1 Incorporating Known Features into Map-Building

Position-based navigation is directly aided by providing the robot with extra information on its world-frame location, especially when far from its starting point. This can be achieved by providing it with the whereabouts of some *known* features in the environment as it sets out.

To initialise a known feature, an image patch representing it must be obtained (in our experiment by wheeling the robot to near the objects chosen and "taking a picture" with its cameras), and its world position measured. In an environment with a known ground-plan such as a factory, this could easily be obtained. A slight problem arises here because we are enforcing an abstracted view of the true position of a feature on the system, as was mentioned in Section 5.4.1: it is difficult to specify the exact location of a landmark in the same way as the robot would see it. With careful choice of feature, however, we can hope to get very close.

The measured feature location is initialised directly into the estimated state vector as the coordinates $\mathbf{y}_i$ of a feature i. To tell the system that the location of this feature is perfectly known, its covariance $\mathtt{P}_{y_i y_i}$ is set with all elements equal to zero, along with the cross-covariances between the feature state and that of the vehicle and other features. In prediction and measurement updates, the filter copes with these perfectly known landmarks with no difficultly: once initialised, they can be treated in exactly the same way as automatically acquired, uncertain features. In normal filter updates, as should correctly occur a perfect feature never becomes tangled with the other estimates, in that the covariance

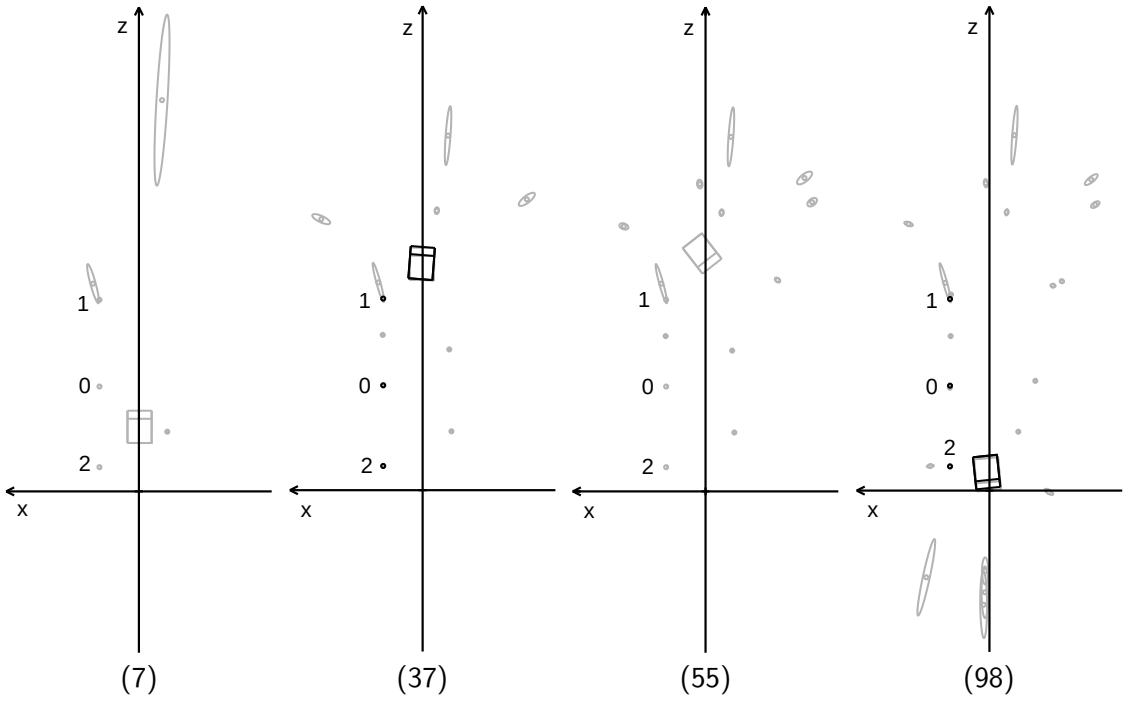


Figure 7.7: Automatic position-based navigation with 3 known features (0, 1 and 2). High localisation accuracy can now be achieved over a wider range of robot movement.

elements relating to it remain at zero throughout.

There have of course been many approaches to robot localisation using known beacons. Sometimes these are artificially placed into an environment, such as GPS satellites or barcodes readable by a laser scanner, and sometimes they are naturally occurring features which are known to the robot in the form of a prior map. When all landmarks used for navigation are in known positions, the localisation problem becomes relatively simple, and one which has been well tackled [27]. In the approach in this section, we do not propose to form a full map of known landmarks, but just show that a small number can be seamlessly incorporated into an automatically-generated map of uncertain features to improve localisation performance in an easy step.

Experiment with Known Landmarks

An autonomous experiment was conducted where the robot made a similar movement to that in the experiment of Section 7.3, but now with 3 known features in the map as it set out. These lay to one side of the corridor, and are labelled as 0, 1 and 2 in the pictures of Figure 7.7 showing the progress of the experiment.

In just the same way that in the experiment of Section 7.3 the automatic feature-choice criterion selected features not measured for a long time whenever possible, in this experiment the known features were selected as soon as they became visible, showing the drift which was occurring in the robot's estimation relative to the world frame. The benefit of the known features was to improve world-frame localisation accuracy when the robot was a

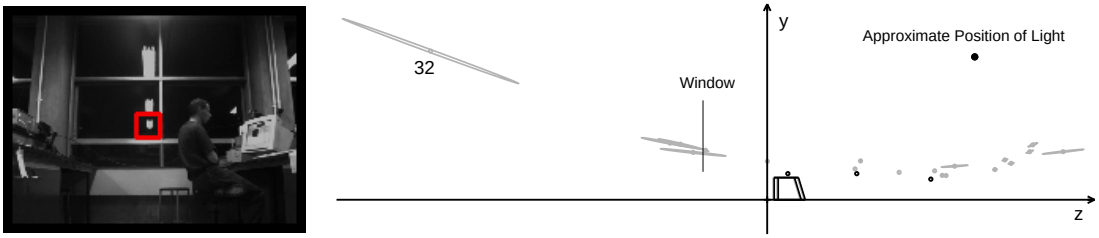


Figure 7.8: A virtual feature: 32 is a reflection in a window of an overhead light. Its position in the map lies outside of the laboratory, but it still acts as a stable landmark.

long way from its origin. At step (37), when the robot was at it farthest distance from the origin, its ground-truth location was measured. The true and estimated locations and the estimate covariance were:

$$\mathbf{x}_v = \begin{pmatrix} 5.83 \\ 0.01 \\ -0.01 \end{pmatrix} \quad , \quad \hat{\mathbf{x}}_v = \begin{pmatrix} 5.81 \\ 0.01 \\ -0.02 \end{pmatrix} \quad , \quad \mathtt{P}_{xx} = \begin{bmatrix} 0.00022 & 0.00004 & 0.00000 \\ 0.00004 & 0.00010 & 0.00005 \\ 0.00000 & 0.00005 & 0.00007 \end{bmatrix} .$$

This accuracy, to within a couple of centimetres, vastly improves on that achieved at a similar position in the experiment of Section 7.3 with no known features (step (124) in Figure 7.6). It can also be seen that automatic features initialised when the robot is in this position are much more certain now: the features at the far end of the corridor (high z) in Figure 7.7 have much smaller ellipses than those in Figure 7.6. Features near to known features become very well-known themselves.

Something interesting arising in this experiment (coincidental to the use of known features) was the use of a virtual feature as a landmark. The experiment was carried out at night under artificial lighting, and feature 32 in the automatically-generated map was a reflection of one of the lights in the large window lying about 1.4m behind the robot starting point, as seen in Figure 7.8 which shows the feature and the state estimated at the end of the robot's run (step 121), when it had returned close to its starting point, from a side-on point of view. The light in question of course actually lay to the front of the robot, since its position estimate in Figure 7.8 is outside of the laboratory: however, its reflection in the planar window proves to be an equally valid feature itself, having a fixed 3D location in the world as the robot moves.

7.4.2 Finding Areas of Free Space

The sparse map of features generated by the map-building system is intended purely as an aid to localisation rather than as a way to identify navigable areas and obstacles, but it was interesting to see how dense a map of features could be generated in an autonomous experiment where a deliberate attempt was made to initialise a lot of them.

Building a Dense Map of Landmark Features

Figure 7.9 shows the end result of an experiment where the the robot ran repeatedly up and down over a distance of around two metres attempting to generate a dense map. For this

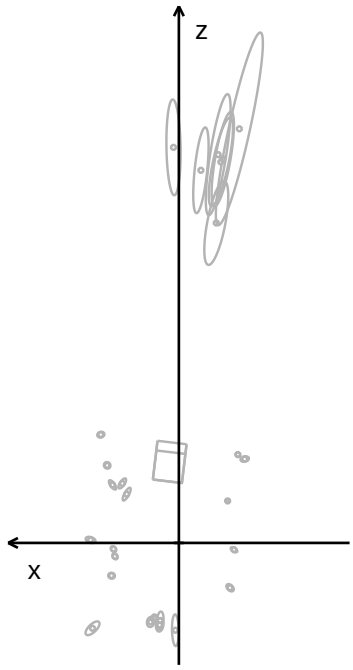


Figure 7.9: A dense map of features automatically generated by the robot while making repeated backward and forward movements in a small area.

purpose, the map-maintenance approach of Section 7.1 was altered. Rather than finding new features only when less than two were visible, features were initialised whenever all the currently visible features all had V_S (the parameter introduced in Section 6.3 describing the uncertainty in relative feature/robot position) scores below the value $V_{S(\text{new})}$ of newly-initialised landmarks. That is to say, if the parts of the map currently visible were well known and consistent, some new features were initialised. This criterion tends to build up features indefinitely even in areas from which good landmarks are already visible; as such, it it not suitable in ordinary navigation because maps become over-dense.

The map generated after 81 movement steps is shown in Figure 7.9. In the region near to the origin, where the robot has been moving backwards and forwards, the shape of the end of the corridor-like bay can clearly be seen. The features there are parts of various objects below, on and above the benches at the sides of the bay, so they do not appear in perfect rows, but the outline is clear. The other cluster of features is at the far end of the corridor, corresponding to objects on the shelves there. There is a tendency with this map-maintenance method for features to be initialised in the same place repeatedly, and there are many overlapping features in this region.

Other Methods for Finding Free Space

Making a dense map of landmark features as above with the aim of identifying free space areas has two clear problems:

- The map is still not dense enough to be reliable. Where there is a gap between points, it is not known whether this is an empty area, or just a region of an obstacle with no

outstanding features.

- Serially fixating on single points with the active head is a very time-consuming way make a dense map.

Any feature-based method for detecting obstacles and free space seems fraught with danger, due to the limitations of granular features, detected in "regions of interest" which do not necessarily correspond to structurally relevant objects. Our system for instance would never find a feature in the middle of a blank wall, and neither would a system making more efficient and dense point maps with the aim of determining free space, such as that of Beardsley *et al.* [6]. How do humans cope with such obstacles? It is probably true that the greater visual resolution (both spatial and dynamic) of the human eye means that it can see features on most walls which would appear blank to a video camera, but still we would expect to be able to identify a wall as such even if it was extremely homogeneous. Once some clue, such as a characteristic reflection, had revealed it as a plane, there would be no danger of colliding with it. Mura and Franceschini [66] have used a biologically-inspired scanning retina to detect obstacles in a terrestrial mobile robot.

It would therefore seem that the ability to fit higher-level feature type to scenes will be crucial to enabling dense world mapping. Some work has already looked into such problems in the structure from motion community with particular success recently, [73, 22], where planes and some curved shapes have been fitted to 3D point sets, allowing data reduction and the classification of a whole region as "obstacle" immediately.

Finding obstacles and areas of free space is something which is perhaps not easily incorporated into an active approach where most of the time is spent tracking single pertinent features: Manyika [57] suggested this as the "Achilles Heel" of the methodology with regard to his work with sonar, since a sensor trained on a feature is generally not well-placed to scan the surroundings for potential problem areas. One visual approach, lending itself to a solution which seems to have much in common with the human approach to navigation, is to use cameras which have a wide field of view, but graduated lenses (as used in [78]) which provide high resolution at the central fovea and low resolution in the outer angular ranges. While foveal processing deals with tracking the current feature, another process can run on the outer parts of the field of view to detect obstacles, perhaps with simple algorithms such as a search for rapidly looming objects. Humans are quite capable of interrupting a tracking task to react to something glimpsed "out of the corner of the eye".

Another approach is for the fixating system to take an occasional break to "glance around" and take note of the surroundings, and this is again something that humans do while moving. Crucially, more than the central foveal area of the eye or robot camera must be used here to search the wide field of view available efficiently. In our system, the current localisation uncertainty can be evaluated to decide when is a good time to take time away from feature measurements.

With a robot moving around the ground plane, the question that needs to be answered to enable safe navigation is whether the area to be moved into consists of flat floor or some construction protruding upwards [55, 61]. This simplifies the obstacle detection problem: a region of free space can be confirmed if the ground-plane can be found. If, as with our robot, the head and robot geometry is approximately known, an area-based method such as correlation or optical flow can be used to confirm that the ground-plane can be found where it should be.

Local searches for free-space could be combined with global waypoint-based navigation. If an obstacle is detected along the robot's desired path, one way to avoid it would be to dynamically lay down some new waypoints in the free-space area surrounding it, and then follow these in a path of avoidance. It is more difficult to see how a large map of areas of free space should be maintained: a grid based approach would be memory-intensive over a large area, and more importantly any estimates of the locations of obstacles and free space are subject to the same uncertainties as the estimates of the positions of landmark features. It would be infeasible to treat a large free-space map with the same detail as the landmark features. We are reminded of the active vision proverb of "using the world as its own best memory": although large-scale features, which can be modelled as planes or lines for instance, could potentially become integrated with the landmark feature map, small obstacles should be treated locally, and searched for actively when an area needs to be moved into. The strategies for doing this will certainly be something to consider in work continuing from this thesis.

7.4.3 The Link with Context-Based Navigation

By context-based navigation, we mean making movements in relation to what can be seen in the near surroundings in order to reach local goals. Useful journeys can be formed by combining movements of this kind. In this chapter, the obstacle-avoidance manoeuvres of Section 7.2.1 fall into this category. What is difficult is how exactly to combine contextual movements towards a total goal. With a form of topological map of a known journey, as perhaps humans have, one small movement leads to another, meaning that complex routes can be travelled without absolute position knowledge.

A robot system can, and therefore perhaps should, do better than this. Position-based navigation, with its requirements of accuracy and fast processing of a lot of data, is exactly the sort of thing that computers do very well. Context-based navigation cannot help in all situations: sometimes position-based is better — when striking out into an open region between two known areas for instance. How can the two types of navigation be linked?

A way in which this can be achieved in our landmark-based system is to attach "labels" to certain features, which provide information on their context in the world and how local navigation should proceed with respect to them. We can foresee how this could be performed automatically: for instance, if a free-space-finding vision module reports that an obstacle lies ahead which has a large amount of free-space to the left side, a landmark feature lying on the left obstacle boundary could be labelled as "a feature which must be steered around to the left side". If no features had already been found in this region, an active attempt could be made to initialise one there.

Another way of attaching a label to a feature is to specify the feature at the start of motion, and to affix the label at that stage with human assistance. We will give some examples in the following two autonomous experiments.

Steering a Twisting Course

An autonomous experiment was conceived where two known features were specified on the inside corners of a zig-zag route to be followed along corridors in the laboratory over a total distance of about 12 metres. These features had positions which were known accurately

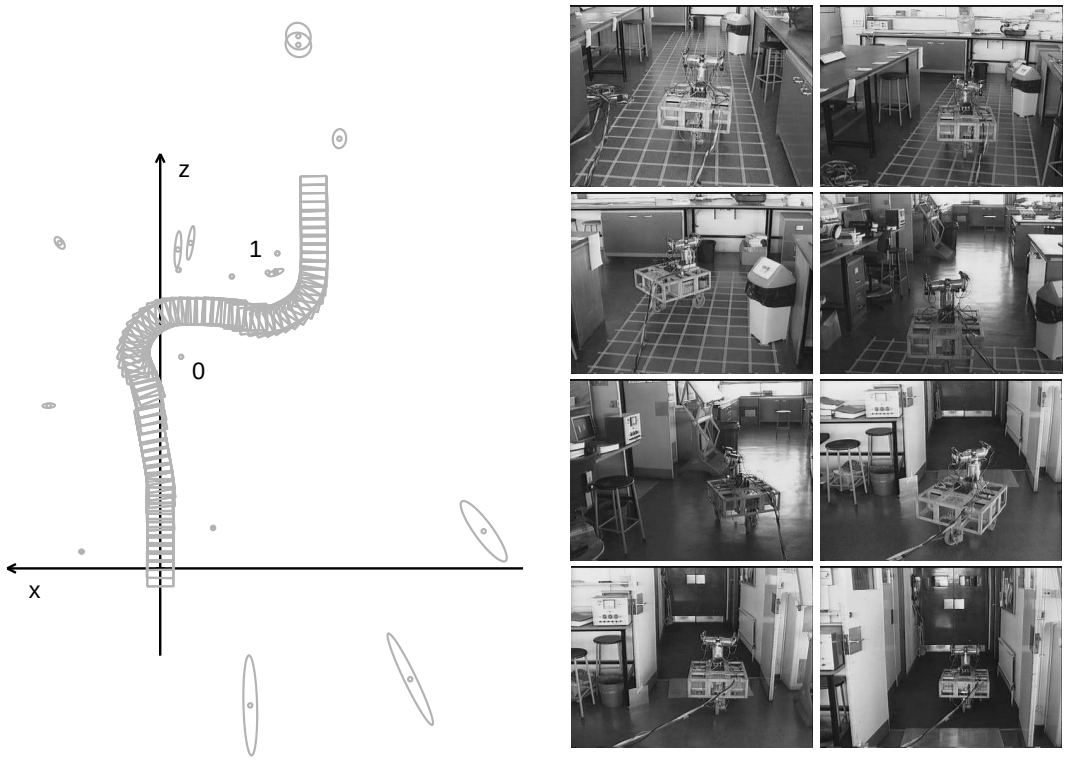


Figure 7.10: The estimated trajectory and frames cut from a video as the robot navigated autonomously around two known landmarks and out of the laboratory door. The navigation knew the locations of features 0 and 1 as prior knowledge, along with information on their status as obstacles.

in the world, and they were initialised into the map as in Section 7.4.1, but with labels attached indicating their importance as lying on the corners of obstacles.

The "journey plan" given to the robot consisted of the following:

1. Go forward to waypoint $(z, x) = (2.0, 0.0)$.
2. Steer to the left of feature 0.
3. Steer to the right of feature 1.
4. Go to waypoint $(z, x) = (8.5, -3.5)$ and stop.

The robot set off, performing map-building and localisation as usual, but steering as described above. The results of the experiment are shown in Figure 7.10, where the estimated trajectory generated is pictured next to stills from a video of the robot safely navigating the course.

Steering control in this experiment involved making transitions between various waypoint-steering and obstacle-steering modes. Switching out of a waypoint-steering stage is trivial, because the waypoint is said to have been reached when the robot is within a certain small distance of it. Making the transition from obstacle-steering is more involved,

but necessary to stop the robot from continuing in a never-ending orbit. The point at which to leave depends on what the robot is going to do next — some obstacles will cause the robot to deflect its trajectory only slightly in a short period of avoidance, while some, such as the inside corner of a 90° turn in a corridor, will require relatively long steering manoeuvres. The solution is to consider the next guidance point in the motion — either waypoint or obstacle — and compare the current steering action with that which the next point would be demanding in the absence of the current obstacle. The transition to the next guidance point should be made when the two instantaneously agree: the current point will have done its job and control can pass to the next. This will only happen when the current obstacle has been cleared sufficiently that the robot can start making for the next waypoint or obstacle-avoidance without danger of hitting it. Figure 7.10 shows the smooth transitions occurring between waypoint and obstacle steering modes which this criterion produces.

Note that in this experiment, steering around the known obstacles took place on a positional basis. The robot steered so as to avoid the known obstacles based on its current position estimate, even before it had first measured them. The automatic feature-selection criterion decided when it was necessary actually to measure the known features, and in the experiments this proved to be as soon as they became visible, in order to lock the robot position estimate down to the world frame. The point when a first measurement of known feature 0 is made can be clearly seen in Figure 7.10 as a small kink in the robot trajectory: actually measuring the feature corrected the robot's drifting position estimate and meant that the steering angle was changed slightly to correct the approach. After this, the obstacle feature was fixated on only when it again became the best measurement to make. Otherwise, attention was paid to improving the map of automatically-acquired features.

Steering Around Closely-Spaced Obstacles

In a very similar experiment, the robot steered a course around two boxes acting as obstacles in known positions. The journey commands this time were:

1. Steer to the left of feature 0.
2. Steer to the right of feature 1.
3. Go to waypoint $(z, x) = (6.0, 0)$ and stop.

The results are shown in Figure 7.11. The safe radius R by which obstacles were to be avoided was set to a value 0.6m which is sufficient for the side of the robot to clear them with some 20cm to spare.

7.4.4 Conclusion

The last section has shown with initial experiments how context-based steering control can be combined with map-building and position-based movement, and we feel that this is the way forward for autonomous robots performing tasks in partially-known or unknown environments. Clearly it is not ideal to need to specify fully known landmarks and obstacles

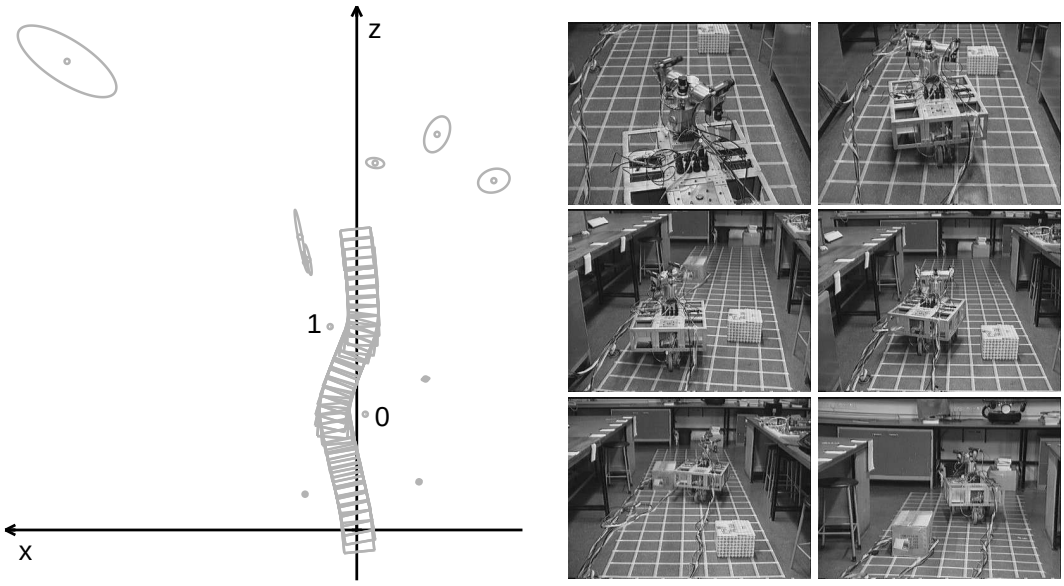


Figure 7.11: The robot tracks and avoids two closely-spaced obstacles in known positions. The active head can be seen fixating on both as it rounds them, and on various other scene features.

as prior information: it is hoped that it will be possible automatically to detect pertinent features to guide manoeuvres.

However, in certain cases, the prior knowledge could have a much looser form which could encode information about a few very important landmarks, but not in a rigidly-mapped way. A feature labelled as an obstacle or trajectory corner could be specified with an uncertainty associated to its location: something like "at about 10 metres ahead, plus or minus a metre, there is an obstacle which must be avoided to the right hand side". This uncertainty would be incorporated into the total state covariance matrix. As the robot approached this feature, it would search the uncertainty region to detect it. Once it had been found, its true world position would become locked into the map and the robot could steer safely around it.

Further, chains of such features could exist, where only the relative positions of each pair is specified as initial knowledge. The nature of this information can be fully encoded in the covariance matrix. Providing prior data of this kind would be much more easily done than providing the true world positions of all the features, but the robot's localisation system would cope and allow it to move along the chain, carefully manoeuvring with respect to each feature once found. This is the kind prior map which could be passed to the task-driven robots of the future, and it seems to have much in common with the way in which humans store route-plans.

8

Conclusions

In one of the first real implementations of navigation using active vision, we have shown that it has a large part to play in the future of autonomous mobile robotics. To conclude this thesis, we will review what it is felt are the main contributions made, and discuss plans for future related work.

8.1 Contributions

The main achievements of this thesis can be outlined as follows:

- An accurate, reliable and consistent algorithm for simultaneous localisation and map-building in unknown environments, using information from measurements of arbitrary features and robot odometry (Chapter 5). The Kalman Filter-based method presented is applicable to many systems, active and passive. The choice of a full-covariance approach has proven its worth for extended periods of navigation, providing in particular the capability to re-measure features after a period of neglect and recover accurate position estimation after periods of drift.

- Good mathematical modelling of vehicle and active head (Chapter 3). These models allowed the system to operate accurately, and included consideration of the uncertainties introduced by the hardware.

- A full robot implementation of the techniques presented for active visual navigation (Chapters 4, 5). Provision of capabilities such as correlation search ellipses, continuous fixated feature tracking and dynamic steering control have allowed ideas to be tested in realistic circumstances.

- Extended and carefully performed experiments in a real environment (Chapters 5, 6, 7). Meaningful experiments were carried out into most aspects of the work, using the robot in a realistic setting with ground-truth comparisons available from an accurate grid.
- An efficient algorithm for updating the Kalman Filter in the case of extended periods of single-feature tracking (Chapter 6). A rearrangement of the bookkeeping of filtering allows updates to be carried out in constant time independent of the number of features in the full map, while information is retained to enable a full unapproximated map update when processing time becomes available.
- An automated strategy for selective fixation on scene features (Chapter 6). The active head is used to best effect to make the feature measurements which do most to improve the quality of map and localisation estimations. The effects of finite active head performance have also been investigated.
- An approach to automatic map maintenance (Chapter 7). Criteria are presented as a basis for deciding when to delete or add features from or to a self-generated map, with the goal of maintaining it in a globally useful and consistent form.
- Experiments in goal-directed navigation (Chapter 7). With the aim of producing a mobile robot which could perform useful tasks, using prior information where available, methods such as incorporating known features into an automatically generated map and steering compound paths through waypoints and around obstacles have been successfully implemented.

8.2 Future Work

The framework put in place by the work in this thesis gives a strong basis for several interesting directions of future work, which will lead to increasing levels of autonomy for mobile robots in useful applications. We feel that work should be concentrated on two main areas:

- The clear benefits demonstrated in this thesis of a full-covariance approach to simultaneous localisation and map-building (Chapter 5) are balanced by the computational complexity of maintaining a map of this form. It is felt that it will be possible to extend the algorithm of Section 6.2, which provides efficient updates for the case of repeated measurements of one feature, to the situation where a certain subset of the total number of features in the map can be kept up to date in real time, while the rest can be updated in one computationally-expensive step when this becomes necessary.

  Full knowledge of the covariances between a certain set of features permits the feature-selection strategies of Section 6.3 to operate and make choices between them. In a particular robot situation, it will only be possible to make measurements of a certain number of features at a given time. If the states and covariances relating these are kept up to date in real time, feature selection can take place as required. It is not necessary, though, to have real-time information about the features which are not measurable. It is hoped that the instructions about how to generically update these

when the time comes will be able to be stored "on the back burner" for a full update to occur when necessary.

This would remove limits (aside from storage space) on the size of global feature maps which could be maintained. For example, an indoor robot moving between rooms would only need to keep real-time information on those features within the current room. When a different room was entered, the robot could stop briefly to perform the full map update step and bring its estimates about the features there up to date.

If successfully formulated, this method could have large benefits in many areas, especially in the structure from motion community in computer vision research, where sequential updating of structure and motion estimates has now frequently been abandoned due to the problems of motion drift in favour of batch approaches, which cannot produce maps in real-time.

Further gains in efficiency could be gained by potentially making approximations about the full covariance matrix. As discussed Section 5.2, it may not be necessary to maintain covariance links between features which are distantly separated in the map (not necessarily in terms of pure distance, but the number of measurements involved in estimating their relative locations). The full covariance matrix `P` would tend to become more diagonal.

- The other main area of future research is more focussed on the problem of active visual navigation. Truly autonomous robots need the capability to detect obstacles and free space, and react by adjusting their movements on the way to a global goal. Free-space representations must be incorporated with the landmark feature map, perhaps with a local dense map for obstacle avoidance, and large-scale features fully incorporated with the landmarks. The use of features other than points, such as lines or planes, in the map-building approach will be considered.

  The idea of contextual navigation will also be investigated further. Automatically attaching labels to map features indicating their status as obstacles will allow the robot to use the steering method already implemented in Chapter 7 to perform avoidance and plan safe paths.

Bibliography

[1] M. Armstrong, A. Zisserman, and R. Hartley. Self-calibration from image triplets. In B. Buxton and R. Cipolla, editors, *Proceedings of the 4th European Conference on Computer Vision, Cambridge*, volume 1, pages 3–16, April 1996.

[2] N. Ayache. *Artificial Vision for Mobile Robots: Stereo Vision and Multisensory Perception*. MIT Press, Cambridge MA, 1991.

[3] R. Bajcsy. Active perception. *Proc.IEEE*, 76(8):996–1005, 1988.

[4] D. H. Ballard and A. Ozcandarli. Eye fixation and kinetic depth. In *Proceedings of the 2nd International Conference on Computer Vision, Tampa*, 1988.

[5] P. H. Batavia, D. A. Pomerleau, and C. E. Thorpe. Overtaking vehicle detection using implicit optical flow. In *Proceedings of the IEEE Intelligent Transportation Systems Conference, Boston, MA*, 1997.

[6] P. A. Beardsley, I. D. Reid, A. Zisserman, and D. W. Murray. Active visual navigation using non-metric structure. In *Proceedings of the 5th International Conference on Computer Vision, Boston*, pages 58–65. IEEE Computer Society Press, 1995.

[7] P. A. Beardsley, A. Zisserman, and D. W. Murray. Navigation using affine structure from motion. In *Proceedings of the 3rd European Conference on Computer Vision, Stockholm*, volume 2, pages 85–96, 1994.

[8] P. A. Beardsley, A. Zisserman, and D. W. Murray. Sequential update of projective and affine structure and motion. Technical Report OUEL report 2012/94, Dept. of Engineering Science, University of Oxford, 1994.

[9] A. Blake, A. Zisserman, and R. Cipolla. Visual exploration of free-space. In A. Blake and A. Yuille, editors, *Active Vision*. MIT Press, Cambridge, MA, 1992.

[10] J.-Y. Bouget and P. Perona. Visual navigation using a single camera. In *ICCV5*, pages 645–652, Los Alamitos, CA, 1995. IEEE Computer Society Press.

[11] K. J. Bradshaw, P. F. McLauchlan, I. D. Reid, and D. W. Murray. Saccade and pursuit on an active head/eye platform. *Image and Vision Computing*, 12(3):155–163, 1994.

[12] R. A. Brooks. A robust, layered control system for a mobile robot. Technical Report A.I. Memo 864, Massachusetts Institute of Technology, September 1985.

[13] R. A. Brooks. Achieving artificial intelligence through building robots. Technical Report A.I. Memo 899, Massachusetts Institute of Technology, May 1986.

[14] R. A. Brooks. Intelligence without representation. In *Workshop on the Foundations of Artificial Intelligence, Dedham, MA*, 1987.

[15] R. A. Brooks and L. A. Stein. Builing brains for bodies. Technical Report A.I. Memo 1439, Massachusetts Institute of Technology, August 1993.

[16] C.M. Brown. Gaze control with interactions and delays. *IEEE Trans. Sys. Man and Cyb.*, 63:61–70, 1990.

[17] T. P. H. Burke. *Design of a Modular Mobile Robot*. PhD thesis, University of Oxford, 1994.

[18] S. A. Cameron and P. J. Probert, editors. *Advanced Guided Vehicles — Aspects of the Oxford AGV Project*. World Scientific, Singapore, 1994.

[19] J. F. Canny. Finding edges and lines in images. Master's thesis, MIT, 1983.

[20] R. Cipolla and A. Blake. The dynamic analysis of apparent contours. In *Proceedings of the 3rd International Conference on Computer Vision, Osaka*, 1990.

[21] J. C. Clarke. *Applications of Sequence Geometry to Visual Motion*. PhD thesis, University of Oxford, 1998.

[22] G. Cross and A. Zisserman. Quadric reconstruction from dual-space geometry. In *Proceedings of the 6th International Conference on Computer Vision, Bombay*, pages 25–31, 1998.

[23] M. Csorba, J. K. Uhlmann, and H. F. Durrant-Whyte. A new approach to simultaneous localization and dynamic map building. In *SPIE Proceedings*, pages 26–36, 1996.

[24] A. J. Davison, I. D. Reid, and D. W. Murray. The active camera as a projective pointing device. In *Proceedings of the 6th British Machine Vision Conference, Birimingham*, pages 453–462, 1995.

[25] L. de Agapito, D. Q. Huynh, and M. J. Brooks. Self-calibrating a stereo head: an error analysis in the neighbourhood of degenerate configurations. In *Proceedings of the 6th International Conference on Computer Vision, Bombay*, pages 747–753, 1998.

[26] F. Du. Navigation in tight clearances using active vision. Technical Report OUEL 2041/94, Department of Engineering Science, University of Oxford, 1994. First year transfer report.

[27] H. F. Durrant-Whyte. Where am I? A tutorial on mobile vehicle localization. *Industrial Robot*, 21(2):11–16, 1994.

[28] S.M. Fairley, I.D. Reid, and D.W. Murray. Transfer of fixation for an active stereo platform via affine structure recovery. In *Proceedings of the 5th International Conference on Computer Vision, Boston*, pages 1100–1105, 1995.

[29] O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In G. Sandini, editor, *Proceedings of the 2nd European Conference on Computer Vision, Santa Margharita Ligure, Italy*, pages 563–578. Springer-Verlag, 1992.

[30] O.D. Faugeras. *Three-Dimensional Computer Vision*. MIT Press, 1993.

[31] N.J. Ferrier. The harvard binocular head. Technical Report 91-9, Harvard Robotics Laboratory, 1991.

[32] C. R. Gallistel. *The Organisation of Learning*. MIT Press, Cambridge MA, 1990.

[33] J. Gluckman and S. K. Nayar. Ego-motion and omnidirectional cameras. In *Proceedings of the 6th International Conference on Computer Vision, Bombay*, pages 999–1005, 1998.

[34] M. Brady H. Wang and C. Bowman. Architectures and algorithms for 3D vision: the parallel Droid system. In S. Cameron and P. Probert, editors, *Advanced Guided Vehicles*, pages 125–139. World Scientific, Singapore, 1994.

[35] C. G. Harris. Tracking with rigid models. In A. Blake and A. Yuille, editors, *Active Vision*. MIT Press, Cambridge, MA, 1992.

[36] C. G. Harris and J. M. Pike. 3D positional integration from image sequences. In *Proc. 3rd Alvey Vision Conference, Cambridge*, pages 233–236, 1987.

[37] C. G. Harris and M. Stephens. A combined corner and edge detector. In *Proc. 4th Alvey Vision Conference, Manchester*, pages 147–151, 1988.

[38] R.I. Hartley. In defence of the 8-point algorithm. In *Proceedings of the 5th International Conference on Computer Vision, Boston*, pages 1064–1070, 1995.

[39] R.I. Hartley, R. Gupta, and T. Chang. Stereo from uncalibrated cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–764, 1992.

[40] J. J. Heuring. *Applications of Computer Vision to Telepresence*. PhD thesis, University of Oxford, 1997.

[41] J. J. Heuring and D. W. Murray. Visual head tracking and slaving for visual telepresence. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1996.

[42] E. Huber and D. Kortenkamp. Using stereo vision to pursue moving agents with a mobile robot. In *IEEE International Conference on Robotics and Automation*, May 1995.

[43] E. Huber and D. Kortenkamp. A behavior-based approach to active stereo vision for mobile robots. To appear in Engineering Applications of Artificial Intelligence Journal, 1998.

[44] S. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. *IEEE Trans. Robotics and Automation*, 12(5):651–669, 1996.

[45] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proceedings of the 4th European Conference on Computer Vision, Cambridge*, pages 343–356, 1996.

[46] T. M. Jochem, D. A. Pomerleau, and C. E. Thorpe. Vision guided lane transition. In *IEEE Symposium on Intelligent Road Vehicles, Detroit, MI*, 1995.

[47] E. Krotkov, M. Hebert, and R. Simmons. Stereo perception and dead-reckoning for a prototype lunar rover. *Autonomous Robots*, 2(4):313–331, 1995.

[48] M. F. Land and J. Horwood. Which parts of the road guide steering? *Nature*, 377:339–340, 1995.

[49] M. F. Land and D. N. Lee. Where we look when we steer. *Nature*, 369:742–744, 1994.

[50] D. Langer, J. K. Rosenblatt, and M. Hebert. A behaviour-based system for off-road navigation. *IEEE Trans. Robotics and Automation*, 10(6):776–782, 1994.

[51] D. Lee and M. Recce. Quantitative evaluation of the exploration strategy of a mobile robot. In *AISB 94 Workshop: Models or behaviours — which way forward for robotics?*, University of Leeds, April 1994.

[52] J. J. Leonard. *Directed Sonar Sensing for Mobile Robot Navigation.* PhD thesis, University of Oxford, 1990.

[53] J. J. Leonard, H. Durrant-Whyte, and I. J. Cox. Dynamic map building for an autonomous mobile robot. *International Journal of Robotics Research*, 11(4):286–298, 1992.

[54] J. J. Leonard and H. F. Durrant-Whyte. *Directed Sonar Navigation.* Kluwer Academic Press, 1992.

[55] F. Li. *Active Stereo for AGV Navigation.* PhD thesis, University of Oxford, 1996.

[56] H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[57] J. Manyika. *An Information-Theoretic Approach to Data Fusion and Sensor Management.* PhD thesis, University of Oxford, 1993.

[58] D. Marr. *Vision.* MIT Press, Cambridge MA, 1982.

[59] S. J. Maybank, A. D. Worrall, and G. D. Sullivan. Filter for car tracking based on acceleration and steering angle. In *Proceedings of the 7th British Machine Vision Conference, Edinburgh*, pages 615–624, 1996.

[60] S.J. Maybank and O. Faugeras. A theory of self-calibration of a moving camera. *International Journal of Computer Vision*, 8(2):123–151, 1992.

[61] J. Mayhew, Y. Zhang, and S. Cornell. The adaptive control of a four-degrees-of-freedom stereo camera head. *Phil. Trans. Royal Society London B*, 337:315–326, 1992.

[62] P. F. McLauchlan. Horatio: Libraries for vision applications. Technical Report OUEL 1967/92, Dept. Engineering Science, University of Oxford, October 1992.

[63] P. F. McLauchlan and D. W. Murray. A unifying framework for structure and motion recovery from image sequences. In *Proceedings of the 5th International Conference on Computer Vision, Boston*. IEEE Computer Society Press, 1995.

[64] P.F. McLauchlan and D.W. Murray. Active camera calibration for a head-eye platform using a variable state-dimension filter. Accepted for PAMI, 1994.

[65] H. Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical Report CMU-RI-TR-3, Carnegie-Mellon University, Robotics Institute, September 1980.

[66] F. Mura and N. Franceschini. Obstacle avoidance in a terrestrial mobile robot provided with a scanning retina. In *Proceedings of the 1996 Intelligent Vehicles Symposium*, pages 47–52, 1996.

[67] D. W. Murray, P. F. McLauchlan, I. D. Reid, and P. M. Sharkey. Reactions to peripheral image motion using a head/eye platform. In *Proceedings of the 4th International Conference on Computer Vision, Berlin*, pages 403–411, Los Alamitos, CA, 1993. IEEE Computer Society Press.

[68] D. W. Murray, I. D. Reid, and A. J. Davison. Steering and navigation behaviours using fixation. In *Proceedings of the 7th British Machine Vision Conference, Edinburgh*, pages 635–644, 1996.

[69] U. Nehmzow. Animal and robot navigation. In *The Biology and Technology of Intelligent Autonomous Agents*. Springer Verlag, 1993.

[70] K. Pahlavan, T. Uhlin, and J-O. Eklundh. Integrating primary ocular processes. In *Proceedings of the 2nd European Conference on Computer Vision, Santa Margharita Ligure, Italy*, pages 526–541, 1992.

[71] J. M. Pichon, C. Blanes, and N. Franceschini. Visual guidance of a mobile robot equipped with a network of self-motion sensors. In *SPIE Conference on Mobile Robots 4*, pages 44–53, 1990.

[72] D. Pomerleau. *Neural Network Perception for Mobile Robot Guidance*. Kluwer Academic Publishers, 1993.

[73] P. Pritchett and A. Zisserman. Wide baseline stereo matching. In *Proceedings of the 6th International Conference on Computer Vision, Bombay*, pages 754–760, 1998.

[74] V. S. Ramachandran. Visual perception in people and machines. In A. Blake, editor, *A.I. and the Eye*, chapter 3. Wiley and Sons, 1990.

[75] I. D. Reid and P. A. Beardsley. Self alignment of a binocular robot. In *Proceedings of the 6th British Machine Vision Conference, Birimingham*, pages 443–452, 1995.

[76] I. D. Reid and D. W. Murray. Tracking foveated corner clusters using affine structure. In *Proceedings of the 4th International Conference on Computer Vision, Berlin*, pages 76–83, Los Alamitos, CA, 1993. IEEE Computer Society Press.

[77] I. D. Reid, D. W. Murray, and K. J. Bradshaw. Towards active exploration of static and dynamic scene geometry. In *IEEE International Conference on Robotics and Automation*, pages 718–723, San Diego, May 1994.

[78] S. Rougeaux and Y. Kuniyoshi. Robust real-time tracking on an active vision head. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[79] S. M. Rowe and A. Blake. Statistical background modelling for tracking with a virtual camera. In *Proceedings of the 6th British Machine Vision Conference, Birimingham*, 1995.

[80] W. Rucklidge. Efficient guaranteed search for gray-level patterns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 717–723, 1997.

[81] S. Scheding, G. Dissanayake, E. M. Nebot, and H. F. Durrant-Whyte. Slip modelling and aided inertial navigation of an LHD. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1904–1909, 1997.

[82] S. Scheding, E. M. Nebot, M. Stevens, H. F. Durrant-Whyte, J. Roberts, P. Corke, J. Cunningham, and B. Cook. Experiments in autonomous underground guidance. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1898–1903, 1997.

[83] C. Schmid and A. Zisserman. Automatic line matching across views. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 666–672, 1997.

[84] P. M. Sharkey, D. W. Murray, S. Vandevelde, I. D. Reid, and P. F. McLauchlan. A modular head/eye platform for real-time reactive vision. *Mechatronics*, 3(4):517–535, 1993.

[85] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[86] E. Shilat, M. Werman, and Y. Gdalyahu. Ridge's corner detection and correspondence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 976–982, 1997.

[87] S. Smith. A new class of corner finder. In *Proceedings of the 3rd British Machine Vision Conference, Leeds*, pages 139–148, 1992.

[88] A. Stevens, M. Stevens, and H. F. Durrant-Whyte. OxNav: Reliable autonomous navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2607–2612, 1995.

[89] H. W. Stone. Mars pathfinder microrover: A low-cost, low-power spacecraft. In *Proceedings of the 1996 AIAA Forum on Advanced Developments in Space Robotics, Madison, WI.*, 1996.

[90] P. Sturm. Critical motion sequences for monocular self-calibration and uncalibrated euclidean reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1100–1105, 1997.

[91] J. I. Thomas, A. Hanson, and J. Oliensis. Understanding noise: The critical role of motion error in scene reconstruction. In *Proceedings of the 4th International Conference on Computer Vision, Berlin*, 1993.

[92] J. I. Thomas, A. Hanson, and J. Oliensis. Refining 3D reconstructions: A theoretical and experimental study of the effect of cross-correlation. *Computer Vision, Graphics, and Image Processing*, 60(3):359–370, 1994.

[93] J. I. Thomas and J. Oliensis. Automatic position estimation of a mobile robot. In *IEEE Conference on AI Applications*, pages 438–444, 1993.

[94] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization approach. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[95] P. H. S. Torr, A. W. Fitzgibbon, and A. Zisserman. Maintaining multiple motion model hypotheses over many views to recover matching and structure. In *Proceedings of the 6th International Conference on Computer Vision, Bombay*, pages 485–491, 1998.

[96] P. H. S. Torr and A. Zisserman. Robust computation and parameterization of multiple view relations. In *Proceedings of the 6th International Conference on Computer Vision, Bombay*, pages 727–732, 1998.

[97] R Y Tsai and T S Huang. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6:13–27, 1984.

[98] H. Wang and J. M. Brady. Corner detection for 3D vision using array processors. In *Proc. BARNAIMAGE-91, Barcelona*. Springer-Verlag, 1991.

[99] J. Weng, T. S. Huang, and N. Ahuja. Motion and structure from two perspective views: Algorithms, error analysis and error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):451–476, 1989.

[100] P. Whaite and F. P. Ferrie. Autonomous exploration: Driven by uncertainty. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):193–205, 1997.

[101] C. S. Wiles, A. Maki, N. Matsuda, and M. Watanabe. Hyper-patches for 3D model acquisition and tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1074–1080, 1997.

[102] C. Zeller and O. Faugeras. Applications of non-metric vision to some visual guided tasks. Technical Report INRIA Rapport de recherche 2308, INRIA Sophia-Antipolis, 1994.

[103] Z. Zhang and O. Faugeras. *3D Dynamic Scene Analysis*. Springer-Verlag, 1992.