AUTONOMOUS LOCALISATION OF ROVERS FOR FUTURE PLANETARY EXPLORATION

A. Bajpai

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey

Surrey Space Centre
Faculty of Engineering and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, U.K.

August 2015

© A. Bajpai 2015
Summary

Future Mars exploration missions will have increasingly ambitious goals compared to current rover and lander missions. There will be a need for extremely long distance traverses over shorter periods of time. This will allow more varied and complex scientific tasks to be performed and increase the overall value of the missions. The missions may also include a sample return component, where items collected on the surface will be returned to a cache in order to be returned to Earth, for further study. In order to make these missions feasible, future rover platforms will require increased levels of autonomy, allowing them to operate without heavy reliance on a terrestrial ground station. Being able to autonomously localise the rover is an important element in increasing the rover’s capability to independently explore.

This thesis develops a Planetary Monocular Simultaneous Localisation And Mapping (PM-SLAM) system aimed specifically at a planetary exploration context. The system uses a novel modular feature detection and tracking algorithm called hybrid-saliency in order to achieve robust tracking, while maintaining low computational complexity in the SLAM filter. The hybrid saliency technique uses a combination of cognitive inspired saliency features with point-based feature descriptors as input to the SLAM filter. The system was tested on simulated datasets generated using the Planetary, Asteroid and Natural scene Generation Utility (PANGU) as well as two real world datasets which closely approximated images from a planetary environment. The system was shown to provide a higher accuracy of localisation estimate than a state-of-the-art VO system tested on the same data set.

In order to be able to localise the rover absolutely, further techniques are investigated which attempt to determine the rover’s position in orbital maps. Orbiter Mask Matching uses point-based features detected by the rover to associate descriptors with large features extracted from orbital imagery and stored in the rover memory prior the mission launch. A proof of concept is evaluated using a PANGU simulated boulder field.

Keywords: Simultaneous Localisation And Mapping, Planetary Guidance Navigation and Control, Feature Detection and Tracking, Autonomous Localisation, Rover Autonomy

Email: a.bajpai@surrey.ac.uk
Acknowledgements

The last several years spent completing my Ph.D have been both challenging and illuminating. Ultimately the experience has been extremely rewarding, and I’m glad to have had the opportunity to expand my analytical and research skills and to grow and learn so much over the course of my studies. Of course, while the work presented in this thesis is the product of one person there are numerous people without whom it would never have reached completion. I would like to take the opportunity to extend my sincerest gratitude to a few important people to whom I and this work owe a great deal.

First and foremost I would like to thank my supervisor, Professor Yang Gao, for offering me the chance to complete a Ph.D. with the STAR Lab group at the Surrey Space Centre. It has been an honour to be able to contribute to the research efforts at such an esteemed centre. Without the benefit of Professor Gao’s experience and knowledge I would not have been able to complete the work in this thesis. Under her supervision I was constantly pushed to achieve a high standard and given the support to reach it. Thanks also to my co-supervisor Professor Sir Martin Sweeting for his insights and comments. In spite of his great responsibilities, any queries or comments regarding my progress were always answered promptly and the responses were of great help.

I would like to thank my parents for all of their love and support. Throughout my academic career they have been enthusiastic cheerleaders, offering encouragement and advice as well as immeasurable patience. Thanks also to my elder sister, my one time nemesis but now a close confidante. Thank you for setting a great example for me and leading the way. Thanks are also due to my cousin Neil, whose own academic achievements served as an inspiration to me in pursuing post-graduate studies. I would also like to thank my grandfather, whose excitement at my achievements and constant pride gave me something to live up to. Thanks also to Melissa, for your support and love.

During the course of my Ph.D. I was afforded the chance to attend the International Astronautical Congress in Beijing in order to present my work. This was made possible by the British Interplanetary Society, who sponsored my attendance. It was a truly once in a life time opportunity, and I would like to thank Alistair Scott and the other members who trusted me to represent British space research at such a prestigious event.

I wish to thank the European Community’s Seventh Framework Programme (FP-7: FASTER Grant agreement no 284419) for providing me with the necessary funding to undertake this research degree.

And finally, last but not least, I would like to thank my friends and colleagues at the SSC and STAR Lab, including Dr. Said Al-Milli, Dr. Affan Shaukat, Dr. Savan Chaniyara, Dr. Brian Yeomans, Guy Burroughes and Conrad Spiteri. Our various formal and informal discussions have always proven to be useful and enjoyable. To all my cousins, aunts and uncles in the United States and India, thank you for all the support and well-wishes.
List of Acronyms

Declaration

1 Introduction
   1.1 Background .................................................. 1
   1.2 Motivation .................................................... 4
   1.3 Contribution .................................................. 6
   1.4 Thesis Structure ............................................. 7

2 Literature Survey
   2.1 Planetary Localisation ....................................... 9
   2.2 Simultaneous Localisation and Mapping .................... 12
      2.2.1 Overview of SLAM ....................................... 12
      2.2.2 Control Signal & Vehicle Model ....................... 13
      2.2.3 Feature Detection & Measurement Model ............... 13
      2.2.4 SLAM Filters .............................................. 15
         2.2.4.1 Extended Kalman Filter .......................... 19
         2.2.4.2 Extended Information Filter ...................... 21
         2.2.4.3 FastSLAM ............................................. 23
         2.2.4.4 Summary of SLAM Filters ......................... 25
## 2.3 Feature Detection

- **2.3.1 Point Based Feature Detection**
  - **2.3.1.1 Scale Invariant Feature Transform**
  - **2.3.1.2 Speeded-Up Robust Features**
  - **2.3.1.3 Binary Robust Invariant Scalable Keypoints**
  - **2.3.1.4 Outlier Rejection**
- **2.3.2 Salient Feature Detection**
  - **2.3.2.1 Hou Saliency**
  - **2.3.2.2 Rudinac Saliency**
  - **2.3.2.3 Thresholding Saliency Heatmaps**
  - **2.3.2.4 Matching and Tracking of Salient Features**

## 2.4 Orbiter Image Localisation

## 3 Planetary Monocular Simultaneous Localisation And Mapping

- **3.1 Introduction**
- **3.2 Structure of PM-SLAM**
- **3.3 Feature Detection and Tracking**
  - **3.3.1 Introduction**
  - **3.3.2 Visual SLAM Using Point-Based Features**
  - **3.3.3 Visual SLAM Using Saliency-Based Features**
  - **3.3.4 Hybrid-Salient Feature Detection and Tracking**
  - **3.3.5 Evaluation of Hybrid-Salient Features**
- **3.4 Monocular Vision-Based Depth Perception**
- **3.5 Implementation**
- **3.6 Experimental Results and Analysis**
  - **3.6.1 Performance Objectives**
  - **3.6.2 Validation Datasets**
  - **3.6.3 Results and Analysis**
- **3.7 Conclusions and future work**
# 4 Orbiter Localisation

4.1 Introduction .................................................. 91
4.2 System Overview ............................................. 92
4.3 Elevation and Inverse Direct Depth ......................... 94
4.4 Orbiter Masking ................................................ 95
   4.4.1 Cylinder Model ........................................ 101
   4.4.2 Dome Model ........................................... 102
   4.4.3 Mask Occlusions ....................................... 105
4.5 Mask Matching ................................................ 107
   4.5.1 Projecting Points onto a Cylinder .................. 109
   4.5.2 Projecting Points onto a Dome ....................... 112
   4.5.3 Point Cloud Matching and Transformation .......... 113
4.6 Experimental Results ......................................... 119
4.7 Conclusions and Future Work ............................... 123

# 5 Summary and Further Work

5.1 Thesis summary ............................................... 125
5.2 Future work ................................................... 127

# Appendices

A Experimental Datasets ........................................ 131
   A.1 PANGU D-shaped localisation dataset .................. 131
   A.2 West Wittering localisation dataset .................... 132
   A.3 SEEKER localisation dataset ............................ 133
   A.4 SEEKER MOTA dataset .................................. 134
   A.5 PANGU straight line traverse ............................ 134

B PM-SLAM Software Architecture ............................. 137
   B.1 Package Structure ....................................... 137
   B.2 PMSLAM-filters .......................................... 138
      B.2.1 SLAMNode ........................................... 138
      B.2.2 SLAMFilterFactory ................................. 138
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.2.3 SLAMFilter</td>
<td>139</td>
</tr>
<tr>
<td>B.2.4 EKFFilter</td>
<td>139</td>
</tr>
<tr>
<td>B.2.5 EIFFilter</td>
<td>139</td>
</tr>
<tr>
<td>B.2.6 FastSLAM</td>
<td>139</td>
</tr>
<tr>
<td>B.2.7 Map</td>
<td>139</td>
</tr>
<tr>
<td>B.2.8 MapFeature</td>
<td>139</td>
</tr>
<tr>
<td>B.2.9 Features3D</td>
<td>140</td>
</tr>
<tr>
<td>B.3 PMSLAM-HybridFeatureDetection</td>
<td>140</td>
</tr>
<tr>
<td>B.3.1 HybridFeatureMatcher</td>
<td>141</td>
</tr>
<tr>
<td>B.3.2 BlobExtractor</td>
<td>141</td>
</tr>
<tr>
<td>B.3.3 FeatureMatcher</td>
<td>141</td>
</tr>
<tr>
<td>B.3.4 DepthExtractor</td>
<td>141</td>
</tr>
<tr>
<td>B.3.5 SaliencyMapFactory</td>
<td>141</td>
</tr>
<tr>
<td>B.3.6 SaliencyMap</td>
<td>142</td>
</tr>
<tr>
<td>B.3.7 SaliencyMapHou</td>
<td>142</td>
</tr>
<tr>
<td>B.3.8 SaliencyMapRudinac</td>
<td>142</td>
</tr>
<tr>
<td>B.3.9 FeatureExtractorFactory</td>
<td>142</td>
</tr>
<tr>
<td>B.3.10 FeatureExtractor</td>
<td>142</td>
</tr>
<tr>
<td>B.3.11 FeatureExtractorSURF</td>
<td>142</td>
</tr>
<tr>
<td>B.3.12 FeatureExtractorSIFT</td>
<td>143</td>
</tr>
<tr>
<td>B.3.13 FeatureExtractorBRISK</td>
<td>143</td>
</tr>
<tr>
<td>B.3.14 DirectDepth</td>
<td>143</td>
</tr>
<tr>
<td>B.4 PMSLAM-Services</td>
<td>143</td>
</tr>
<tr>
<td>B.4.1 ExtractBoundingBox</td>
<td>143</td>
</tr>
<tr>
<td>B.4.2 ExtractFeatureDepths</td>
<td>144</td>
</tr>
<tr>
<td>B.4.3 ExtractFeatureMatches</td>
<td>144</td>
</tr>
<tr>
<td>B.4.4 HybridFeatureMatch</td>
<td>144</td>
</tr>
<tr>
<td>B.4.5 SLAMFilterService</td>
<td>145</td>
</tr>
<tr>
<td>B.5 PMSLAM-Messages</td>
<td>145</td>
</tr>
<tr>
<td>B.5.1 BoundingBox</td>
<td>145</td>
</tr>
<tr>
<td>B.5.2 Feature3D</td>
<td>146</td>
</tr>
</tbody>
</table>
2.5.3 ImageFeature .................................................. 146

2.6 PMSLAM-router ................................................. 146
    2.6.1 Router ..................................................... 146

2.7 Other PM-SLAM Modules ...................................... 147
    2.7.1 PM-SLAM data ............................................ 147
    2.7.2 PM-SLAM GUI ............................................. 147
    2.7.3 PM-SLAM launch .......................................... 147
    2.7.4 PM-SLAM results ......................................... 147

Bibliography .......................................................... 149
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Selected history of Mars exploration mission types</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Panoramic images of sites visited by the Mars Exploration Rover Spirit (NASA/JPL/Cornell)</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>Overview of a SLAM system architecture. The white boxes indicate inputs to the SLAM system, whereas the grey boxes show elements of the SLAM filter, which performs the pose and map estimation.</td>
<td>12</td>
</tr>
<tr>
<td>2.2</td>
<td>Movement of the rover having been given a control vector. The axes in the bottom left corner represent the origin of the map and all co-ordinates are relative to it. The red circle represents the rover at time $t-1$ before the control signal and the blue circle represents rover at time $t$. The solid lines in the circles represent the heading angles.</td>
<td>14</td>
</tr>
<tr>
<td>2.3</td>
<td>Measurement model. The red circle represents the rover, and the red line represents the heading angle. The axes in the bottom left corner represent the origin of the map and all co-ordinates are relative to them.</td>
<td>16</td>
</tr>
<tr>
<td>2.4</td>
<td>Stages of full SLAM iteration. (a) Rover moves forward from bottom of the figure. Two previously encountered features are shown as black stars. The white stars are newly observed features. (b) SLAM predict step produces an estimate of the rover position in red. (c) Rover uses estimated position to predict the position of previously observed features. These predictions are shown as red stars. (d) SLAM update step uses re-observed features to improve rover position estimate, shown in blue. (e) Using the improved estimate the SLAM augment step adds newly observed features to its map.</td>
<td>18</td>
</tr>
<tr>
<td>2.5</td>
<td>Structure of FastSLAM particles [100].</td>
<td>25</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>2.6</td>
<td>An example of image feature detection and matching using the SIFT algorithm. Two consecutive images are shown side-by-side from a simulated rover traverse. The red dots indicate detected SIFT points, and the red lines represent the co-registration of points between the two images.</td>
<td></td>
</tr>
<tr>
<td>2.7</td>
<td>An example of image feature detection and matching using the SURF algorithm. Two consecutive images are shown side-by-side from a simulated rover traverse. The red dots indicate detected SURF points, and the red lines represent the co-registration of points between the two images.</td>
<td></td>
</tr>
<tr>
<td>2.8</td>
<td>An example of image feature detection and matching using the BRISK algorithm. Two consecutive images are shown side-by-side from a simulated rover traverse. The red dots indicate detected BRISK points, and the red lines represent the co-registration of points between the two images.</td>
<td></td>
</tr>
<tr>
<td>2.9</td>
<td>(a) SURF pairs matched between two images before outlier rejection. Two consecutive images are shown side-by-side separated by a white line. Detected feature keypoints are shown as red circles, and correlated points between the two images are connected by red lines. (b) SURF pairs after outliers have been rejected using the RANSAC algorithm described in [32]. The number of correlations has been reduced by outlier rejection, but the remaining co-registered points are more accurate.</td>
<td></td>
</tr>
<tr>
<td>2.10</td>
<td>Output of salient blob detection using Hou algorithm on PANGU generated images. The left image is the original, and the right is the image with salient contours detected by the Hou algorithm outlined in red.</td>
<td></td>
</tr>
<tr>
<td>2.11</td>
<td>Output of salient blob detection using Rudinac algorithm on PANGU generated images. The left image is the original, and the right is the image with salient contours detected by the Rudinac algorithm outlined in red.</td>
<td></td>
</tr>
<tr>
<td>2.12</td>
<td>(a) Grayscale heatmap produced by salient feature extraction. (b) Binarised blob mask produced from heatmap using Otsu’s method [77].</td>
<td></td>
</tr>
<tr>
<td>2.13</td>
<td>Salient features in sequential images tracked using k-NN search. Detection is performed on the left image, and then on the right image after the camera has moved. The bounding boxes represent features, and the numbers represent the tracked id across the subsequent images.</td>
<td></td>
</tr>
<tr>
<td>2.14</td>
<td>A HiRISE image of Aureum Chaos (NASA/LPL).</td>
<td></td>
</tr>
</tbody>
</table>
2.15 Planetary Constellation Matching localisation system [91]. (a) The rover traverses from the starting point at the bottom of the map. The green line represents the planned path. The red line represents the actual path traversed by the rover, including deviations due to obstacle detection and also wheel slip. The blue line represents a localisation estimate from a VO or SLAM system. After a fixed traverse (e.g. 15 metres), the rover takes a panoramic image of its surroundings and identifies large features (>1m diameter), which have been highlighted in green. (b) shows the search area in the orbital map, centred on the rover’s estimated location which is also searched for large features, which have been highlighted in blue. (c) The top graph shows a top down view of the rover’s local feature map, derived from the panoramic imagery with the rover position (the map origin) represented by the black dot. The bottom graph shows a top down view of the global map, derived from the orbital imagery. Corresponding points in these two maps are circled in red, and would be co-registered using an ICP algorithm.

3.1 Block diagram of PM-SLAM ......................................................... 47

3.2 Number of features generated using SURF algorithm with varying initial Hessian size. SURF keypoints are represented by red dots. .................................. 52

3.3 Performance of SLAM using SURF features. The blue line shows the rover pose error increasing with the initial Hessian size, while the red line shows that the average duration of each SLAM step decreases. ........................................... 53

3.4 Hybrid-salient feature detection and tracking data pipeline. .................................. 55

3.5 An example showing each step of hybrid-salient feature tracking: (a) Rectified image from SEEKER dataset, (b) Saliency map, (c) Bounding boxes extracted from saliency map. Bounding boxes that are too small or do not contain enough point-based features have been discarded (the small bounding box on the right of the image), (d) Point-based features extracted from within the bounding boxes, (e) Hybrid-salient features matched against database, with new features in blue, previously observed features in red with their associated feature ID. ........................................... 56

3.6 Demonstration of outlier rejection difference between traditional point-based feature matching and hybrid-salient matching. (a) An example of traditional feature matching between two sequential SEEKER images using SURF and FLANN. Two consecutive images are shown side-by-side. The features are represented by white circles and correspondences between the two images are shown by lines connecting co-registered points between the two images. (b) An example of feature matching between two sequential SEEKER images using Hou and SURF hybrid-saliency implemented in PM-SLAM. Two consecutive images are shown side-by-side. The features are labelled with their associated feature ID, with the bounding boxes of new features shown in blue and previously observed features shown in red. IDs with the same number are matching images across the consecutive images. ........................................... 59
3.7 Steps of annotation tool used to generate feature tracking data for SEEKER dataset. (a) Selecting features in scene. (b) Assigning feature IDs to recorded features. .......................... 62

3.8 Results of hybrid-saliency tracking evaluation using 1136 SEEKER images. (a) Average MOTA for traverse as far as the current traverse step, shown on the x-axis. (b) Time taken to perform detection and matching for each step. ................................................................. 63

3.9 MOTA results for images with differing illumination ............................................. 65

3.10 Simulated illumination variation. (a) (-100) illumination in Image. (b) Unaltered Image. (c) (+100) illumination in Image. ................................................................. 66

3.11 Illustration of image plane and ground plane, with camera parameters (side view). ........................................................................................................... 67

3.12 Illustration of image plane and ground plane, with camera parameters (top-down view). ......................................................................................................... 68

3.13 Graphs showing relation between image-plane y pixel position and real-world (a) y-axis position and (b) x-axis position. .................................................. 70

3.14 Illustration of the differences in error for direct depth y-axis estimates using the centroid and bottom edge of the feature bounding box. The rover assumes a flat ground plane, shown as a dotted line, but the actual terrain has slight variation in height. The boulder appears in the image plane bounded by the green and red lines. The blue line represents a ray from the centre of the bounding box to the rover’s ground plane. The blue and green dots represent the real world positions of the centroid and bottom edge of the image plane bounding box respectively. The rays through these points are continued down to the ground plane, representing the direct depth estimate. The difference between these points on the y-axis represent the depth errors, \( e_c \) and \( e_b \) for the centroid and bottom edge. ................................................................. 71

3.15 Direct depth estimate error for bounding box centroid and bounding box bottom centre. ........................................................................................................... 72

3.16 Example images from: (a) PANGU simulated dataset, (b) WW dataset, (c) SEEKER Atacama desert dataset. ................................................................. 76

3.17 (a) West Wittering trials utilised the SMART rover platform equipped with monocular cameras and DGPS for groundtruthing, (b) SEEKER rover performing trials in Chile (ESA). Both rovers demonstrate the high pitch angle of cameras mounted on planetary platforms. ................................................................. 78

3.18 PM-SLAM cumulative step times on limited hardware setup. .......................... 80

3.19 PM-SLAM loop closure demonstrated using an EKF SLAM filter, with Hou-SURF hybrid-salient features using the PANGU dataset. Traverse is shown in metres. ........................................................................................................... 81
3.20 PM-SLAM results for the PANGU dataset, showing (a) average absolute rover position error for PANGU and (b) the average per-step computation time using different combinations of hybrid-saliency and SLAM filters.

3.21 PM-SLAM heading results for the PANGU dataset showing the absolute rover heading error in radians for each of the combinations of .

3.22 PM-SLAM results for the WW dataset, showing (a) average absolute rover position error for WW and (b) the average per-step computation time using different combinations of hybrid-saliency and SLAM filters.

3.23 PM-SLAM heading results for the West Wittering dataset showing the absolute rover heading error in radians for each of the combinations of .

3.24 Absolute rover position error for SEEKER dataset over a 102 metre traverse, showing dead-reckoning, on-board localisation(VO) and PM-SLAM results.

3.25 PM-SLAM rover position errors for the (a) x (lateral) traverse and the (b) y (forward) traverse components. The dotted lines represent the 3σ error bounds of the filter.

3.26 PM-SLAM rover heading errors. The dotted lines represent the 3σ error bounds of the filter.

3.27 ROS node diagrams showing live examples of PM-SLAM nodes and services. (a) PM-SLAM using an EKF filter with Hou-SURF hybrid-saliency feature detection and tracking. (b) PM-SLAM using an EIF filter with Rudinac-SIFT hybrid-saliency feature detection and tracking.

4.1 System overview, for associating point based feature descriptors with global map features for rover localisation.

4.2 Projection of image point to two different real world points. Point P is the projection to the ground plane and point P′ is the projection of the same image point to a point raised e from the ground.

4.3 Example of extracting features from orbital image using difference of Gaussian and Otsu Thresholding. (a) An example orbiter image generated using PANGU. (b) Blob map generated after applying difference of Gaussians to the orbiter image. (c) Binarised map generated using Otsu’s method on blob map.

4.4 (a) Top down view of boulders on planetary surface. (b) Example of orbiter map, where boulders are stored as points. (c) The rover position is shown as a pink circle. The orbiter map can be used to determine the positions of boulders in the area surrounding the rover. These are shown as green circles, with their size representing the 1m minimum diameter of features visible from orbit. The heading of the rover along with knowledge of the field of view (bounded by the blue lines) allows determination of which boulders should be visible to the rover’s camera.
4.5 (a) The global map, with boulder positions shown as red crosses. The rover is located at the pink circle and has a field of view bounded by the blue lines. (b) Local map, where the boulders outside of the field of view have been removed and the map transformed to the rover’s local perspective. (c) Image showing boulder centroids in the image plane, calculated using the local map and the inverse direct depth algorithm.

4.6 (a) The footprint of each boulder if assumed to be 1 metre in diameter. (b) The boulder footprints transformed to the image plane using inverse direct depth.

4.7 Outline of cylinders representing boulder features mapped onto the image plane using inverse direct depth. The pink circle represents the projected centroid from the map.

4.8 The image mask generated from the cylinder models.

4.9 Profile of the dome model in the y-z plane. The semi-major axis of the cross section semi-ellipse is shown in red.

4.10 Dome representations of boulders drawn onto the image plane.

4.11 The image mask generated from the dome models.

4.12 (a) Feature masks, with overlapping areas highlighted in red. (b) Feature mask for feature 4 including overlap area. (c) Feature mask for feature 3 excluding overlap area.

4.13 Example of matching features using orbiter masks with hybrid-saliency.

4.14 The ray from the camera to the detected point on the model. Points $P_1$ and $P_2$ are points on the line determined from direct depth at two arbitrary elevations.

4.15 Top down view of a cylinder model for a feature, within the rover field of view including a ray through a point feature which intersects the cylinder.

4.16 (a) Example cylinder masks with evenly spaced key points. (b) Points projected onto cylinder model in real world 3-dimensional space.

4.17 (a) Example dome masks with evenly spaced key points. (b) Points projected onto dome model in real world 3-dimensional space.

4.18 The search space within the global map for feature points. The pink circle represents the rover position and the blue line, the bounds of the field of view including errors.

4.19 (a) Point cloud representing the point based features detected in the rovers local field, projected to the height of their matching feature in the global map. (b) Point cloud representing the corresponding feature points in the global map.

4.20 The local point cloud, shown in blue, has been transformed, resulting in the green points. The target points of the global point cloud are shown in red.
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.21</td>
<td>Results of two iterations of computing the rigid transform parameters. In (d) the mean error of the resulting transforms is shown to be decreased.</td>
</tr>
<tr>
<td>4.22</td>
<td>Example image from mask matching PANGU dataset.</td>
</tr>
<tr>
<td>4.23</td>
<td>Simulated boulder positions in red, and rover path in blue.</td>
</tr>
<tr>
<td>4.24</td>
<td>Rover localisation estimate using cylinder model mask matching.</td>
</tr>
<tr>
<td>4.25</td>
<td>Rover localisation estimate using dome model mask matching.</td>
</tr>
<tr>
<td>4.26</td>
<td>Absolute error over traverse using dome and cylinder model mask matching.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1</td>
<td>Modules in the pmslam_filters package. These modules all contribute to the implementation of the SLAM filters. Modules marked with ROS are those that provide an interface with the ROS architecture of PM-SLAM as a whole. Modules marked with SLAM are those that implement the filters and their wrappers. Modules marked with MAP are classes for storing map and feature data.</td>
</tr>
<tr>
<td>B.2</td>
<td>Modules in the pmslam_hybridfeaturedetection package. These modules all contribute to the implementation of Hyrbid Feature Detection. Modules marked with ROS are those that provide an interface with the ROS architecture of PM-SLAM as a whole. Modules marked with SA-LIENCY are those that implement the saliency algorithms and their wrappers. Modules marked with POINT are those that implement the point-based feature detection algorithms and their wrappers. Modules marked with DEPTH implement depth detection algorithms.</td>
</tr>
<tr>
<td>2.3</td>
<td>Services used by PM-SLAM</td>
</tr>
<tr>
<td>2.4</td>
<td>Messages used by PM-SLAM</td>
</tr>
<tr>
<td>2.5</td>
<td>Modules in the pmslam_router package. This module implements the synchronisation of sensor data using ROS.</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2.1</td>
<td>MSL rover drive modes, usage and estimated drive rates [35]</td>
</tr>
<tr>
<td>2.2</td>
<td>Steps of an Extended Kalman Filter SLAM system</td>
</tr>
<tr>
<td>2.3</td>
<td>Steps of an Extended Information Filter SLAM system</td>
</tr>
<tr>
<td>2.4</td>
<td>Steps of an FastSLAM system</td>
</tr>
<tr>
<td>2.5</td>
<td>Comparison of common SLAM filters [90]</td>
</tr>
<tr>
<td>2.6</td>
<td>Datasets used to evaluate ground and aerial feature matching for a</td>
</tr>
<tr>
<td></td>
<td>autonomous terrestrial forest vehicle [46]</td>
</tr>
<tr>
<td>2.7</td>
<td>Results of rover localisation using coregistration of tree stems with tree</td>
</tr>
<tr>
<td></td>
<td>crowns from ground and aerial platforms. The rover pose was estimated</td>
</tr>
<tr>
<td></td>
<td>234 times at intervals of 0.5m [46]</td>
</tr>
<tr>
<td>2.8</td>
<td>Results of rover localisation using coregistration of manually classified</td>
</tr>
<tr>
<td></td>
<td>tree stems and tree crowns from ground and aerial platforms [46]</td>
</tr>
<tr>
<td>2.9</td>
<td>Geometric and texture constraints for feature classification in [16]. H</td>
</tr>
<tr>
<td></td>
<td>represents the distribution of the eigenvalues of the Hessian. E</td>
</tr>
<tr>
<td></td>
<td>represents the distribution of the local entropy</td>
</tr>
<tr>
<td>3.1</td>
<td>MOTA results for detection and tracking using hybrid-saliency on SEEKER</td>
</tr>
<tr>
<td></td>
<td>data</td>
</tr>
<tr>
<td>3.2</td>
<td>MOTA results for tracking using hybrid-saliency on SEEKER data</td>
</tr>
<tr>
<td>3.3</td>
<td>PM-SLAM timings combined with other elements of rover GNC. Each process is</td>
</tr>
<tr>
<td></td>
<td>listed, along with the interval at which it is carried out and the time</td>
</tr>
<tr>
<td></td>
<td>taken each time the process is run. The final column indicates the average</td>
</tr>
<tr>
<td></td>
<td>number of seconds required by each process for a 1cm traverse.</td>
</tr>
<tr>
<td>2.1</td>
<td>ExtractBoundingBox service parameters</td>
</tr>
<tr>
<td>Table</td>
<td>Title</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>2.2</td>
<td>ExtractFeatureDepths service parameters</td>
</tr>
<tr>
<td>2.3</td>
<td>ExtractFeatureMatches service parameters</td>
</tr>
<tr>
<td>2.4</td>
<td>HybridFeatureMatch service parameters</td>
</tr>
<tr>
<td>2.5</td>
<td>SlamFilterService service parameters</td>
</tr>
<tr>
<td>2.6</td>
<td>BoundingBox message definition</td>
</tr>
<tr>
<td>2.7</td>
<td>Feature3D message definition</td>
</tr>
<tr>
<td>2.8</td>
<td>ImageFeature message definition</td>
</tr>
<tr>
<td>Acronym/Abbreviation</td>
<td>Meaning</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------</td>
</tr>
<tr>
<td>AGAST</td>
<td>Adapative and Generic Accelerated Segment Test</td>
</tr>
<tr>
<td>BRIEF</td>
<td>Binary Robust Independent Elementary Features</td>
</tr>
<tr>
<td>BRISK</td>
<td>Binary Robust Invariant Scalable Keypoints</td>
</tr>
<tr>
<td>CFSL</td>
<td>Compact Fast Scanning LIDAR</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>D-GPS</td>
<td>Differential Global Positioning System</td>
</tr>
<tr>
<td>DARCES</td>
<td>Data Aligned Rigidity Constrained Exhaustive Search</td>
</tr>
<tr>
<td>EIF</td>
<td>Extended Information Filter</td>
</tr>
<tr>
<td>EKF</td>
<td>Extended Kalman Filter</td>
</tr>
<tr>
<td>ESA</td>
<td>European Space Agency</td>
</tr>
<tr>
<td>FASTER</td>
<td>Forward Acquisition of Soil and Terrain data for Exploration Rover</td>
</tr>
<tr>
<td>FLANN</td>
<td>Fast Library for Approximate Nearest Neighbour</td>
</tr>
<tr>
<td>FOV</td>
<td>Field Of View</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>GESTALT</td>
<td>Grid-based Estimation of Surface Traversability Applied to Local Terrain</td>
</tr>
<tr>
<td>GNC</td>
<td>Guidance Navigation and Control</td>
</tr>
<tr>
<td>GNSS</td>
<td>Global Navigation Satellite System</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>HFOV</td>
<td>Horizontal Field Of View</td>
</tr>
<tr>
<td>HiRISE</td>
<td>High Resolution Imaging Science Experiment</td>
</tr>
<tr>
<td>ICP</td>
<td>Iterated Closest Point</td>
</tr>
<tr>
<td>IMU</td>
<td>Inertial Measurement Unit</td>
</tr>
<tr>
<td>k-NN</td>
<td>k Nearest Neighbours</td>
</tr>
<tr>
<td>LIDAR</td>
<td>Light Detection And Ranging</td>
</tr>
<tr>
<td>MER</td>
<td>Mars Exploration Rover</td>
</tr>
<tr>
<td>MOTA</td>
<td>Multiple Object Tracking Accuracy</td>
</tr>
<tr>
<td>MRO</td>
<td>Mars Reconnaissance Orbiter</td>
</tr>
<tr>
<td>MSL</td>
<td>Mars Science Laboratory</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautics and Space Administration</td>
</tr>
<tr>
<td>PANGU</td>
<td>Planetary and Asteroid Natural scene Generation Utility</td>
</tr>
<tr>
<td>PCA-SIFT</td>
<td>Principal Components Analysis SIFT</td>
</tr>
<tr>
<td>PM-SLAM</td>
<td>Planetary Monocular Simultaneous Localisation And Mapping</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RANSAC</td>
<td>Random Sample Consensus</td>
</tr>
<tr>
<td>ROI</td>
<td>Region Of Interest</td>
</tr>
<tr>
<td>ROR</td>
<td>Rejection of Outliers by Rotations</td>
</tr>
<tr>
<td>ROS</td>
<td>Robot Operating System</td>
</tr>
<tr>
<td>SFR</td>
<td>Sample Fetch Rover</td>
</tr>
<tr>
<td>SIFT</td>
<td>Scale Invariant Feature Transform</td>
</tr>
<tr>
<td>SLAM</td>
<td>Simultaneous Localisation And Mapping</td>
</tr>
<tr>
<td>SMART</td>
<td>Surrey Mobile Autonomy and Robotics Testbed</td>
</tr>
<tr>
<td>SMPA</td>
<td>Sense Model Plan Act</td>
</tr>
<tr>
<td>STAR Lab</td>
<td>Surrey Technology for Autonomy and Robotics Lab</td>
</tr>
<tr>
<td>SURF</td>
<td>Speeded Up Robust Features</td>
</tr>
<tr>
<td>SVD</td>
<td>Singular Value Decomposition</td>
</tr>
<tr>
<td>VFOV</td>
<td>Vertical Field Of View</td>
</tr>
<tr>
<td>VL</td>
<td>Visual Localisation</td>
</tr>
<tr>
<td>VO</td>
<td>Visual Odometry</td>
</tr>
</tbody>
</table>
Declaration

Research material from the following manuscript has been published or might be published in the following conference/journal proceedings:


1.1 Background

Throughout history there has been a strong interest in the scientific exploration of Mars. The proximity of the red planet to Earth, as well as its similar size make it a useful target for understanding the formation of the solar system and the possible origins of life. With the advent of spacecraft, the opportunity has arisen to more closely make observations of Mars and increase the sum of knowledge about the red planet. The past 50 years has seen a large number of missions focussed on exploring and understanding our planetary neighbour. As technological capability has increased, so have the complexity and ambition of the missions and the platforms they use (Figure 1.1). In the 1960s NASA’s Mariner 4, 6 and 7 flyby missions [96] provided the first opportunity to collect close-up images of the Martian surface. The next generation of spacecraft were designed to insert themselves into orbit around Mars, giving the ability to create higher coverage maps and make observations over longer periods of time. Successes included the Soviet Mars 2 and 3 orbiters [67] and NASA’s Mariner 9 [70]. By 1975 spacecraft technology had evolved to allow a successful lander mission. NASA’s
Chapter 1. Introduction

Figure 1.1: Selected history of Mars exploration mission types.

Viking 1 mission included an orbiter that also deployed a lander to the surface of the planet [66]. Landers offered an opportunity for the first time to make in situ scientific observations and measurements. Landers were limited, however, in that they could only perform their mission tasks at the landing site. The success of the missions was dependent upon careful selection of the landing site and the precision of the landing. If the area where the lander finally settled was devoid of features of scientific interest then the scientific return would not have been enough to justify the expense and complexity of the mission.

The most recent missions have involved the use of planetary rovers. These platforms offer the benefits of making in situ observations and performing site specific experiments, with the added benefit of mobility. The rovers are able to move beyond their initial landing site and explore areas of interest. This not only reduces the burden on the precision of the landing system (effectively increasing potential landing ellipses), but also allows the rovers to have a broader remit, potentially exploring multiple sites, with different geological characteristics. For instance, the Mars Exploration Rovers (MER), Spirit and Opportunity, initially designed with an intended mission lifetime of 90 sols, were able to visit features as diverse as Bonneville crater (Figure 1.2a) [55] and the Columbia Hills (Figure 1.2b) [4] over their operational lifetimes. The Mars Science Laboratory (MSL), has been designed to explore the area around Mount Sharp and observe sedimentation in a difficult terrain.

Future missions will look to expand on the ambition of those that have gone before. The trend in Mars exploration mission design is towards a Sample Fetch Return (SFR)
1.1. Background

Panoramic images of sites visited by the Mars Exploration Rover Spirit (NASA/JPL/Cornell)

Figure 1.2: Panoramic images of sites visited by the Mars Exploration Rover Spirit (NASA/JPL/Cornell)

mission specification. This mission profile would involve a rover landing on the planetary surface and traversing to an area where rock and soil samples of interest could be found. Upon finding these samples, the rover would store them and carry them towards a cache, which would be in the form of a spacecraft that could launch the samples back to Earth. This would allow the samples to be studied directly by scientists and drastically improve current knowledge of Martian geological history. An SFR mission would likely involve long traverses, through differing types of terrain. The rover would have to be able to pinpoint its position with a high degree of accuracy in order to locate useful samples and also to find the cache and deposit them.

A key element in achieving the goals of an SFR mission is increased rover autonomy. Without the rover being able to perform certain tasks without human intervention, it is not feasible for the rover to cover the required traverse distance within a reasonable time.
1.2 Motivation

Both NASA and ESA are planning sample return missions, expected to be launched sometime after 2024. These missions will require an SFR platform to navigate to relevant sites, collect samples from these sites and then return to a capsule in order to launch them back to Earth. In order to achieve this the rover platform has to be able to meet a challenging set of requirements 2:

- The SFR rover will need to complete a 15km straightline traverse over the course of the mission. Taking into account deviations due to terrain and obstacle avoidance, the total traverse would be approximately 20km.

- This traversal would need to be completed within 180 sols. Including planned stoppages and time for collecting samples, the total actual traversal time would be closer to 110 sols.

- In order to achieve this the rover will need to be able to move at a nominal speed of 180m/sol.

- Given an operational window of 2.2 hours per sol (due to power and visibility constraints), the rover would need to move at a speed of 2.2cm/s.

- At the end of each traverse the rover localisation error would need to be below the 5% achieved in ExoMars testing 97 and approaching a design goal of 1% 104.

Current rover navigation systems, such as those on the MERs and MSL, use Visual Odometry (VO) or Visual Localisation (VL). The nature of these techniques requires them to stop every 1.5m - 2m in order to re-evaluate the terrain and hence the traversal speed is limited 14. MSL has a maximum autonomous driving speed of 1.25cm/s 35. On top of this, the relative location estimate produced acquires drift over time, and is especially sensitive to changes in heading direction. Therefore manual re-localisation is required, by sending local imagery back to the ground station, where human operators match the observed terrain against previously acquired orbiter data. Communication with the ground station is currently limited to two windows per sol as signals are relayed
1.2. Motivation

through orbiters. These factors limit the amount of ground that can be covered per sol.

These existing Guidance Navigation and Control (GNC) systems do not allow the rovers to move at a speed which would allow them complete the required objectives of a SFR mission. The goal of this research is to investigate and develop technologies for autonomous localisation of a planetary exploration rover on the Martian surface.

Localisation is only one part of a full GNC system. A typical GNC will include locomotion systems, hazard avoidance and path planning algorithms, which all need to work in concert to ensure a swift and safe traverse. In order to generate hazard maps and perform hazard avoidance path planning a 100 second stop every 15 metres is required. Hazard detection and path planning is required to run for 20 seconds every 0.56 metres (this value is dependent on the angle and field of view of the hazard avoidance cameras, i.e. how far can the rover safely see) [91].

Simultaneous Localisation And Mapping (SLAM) is a robust technique for localisation, that has been developed extensively for terrestrial applications, for both indoor and outdoor robotics. However, the application of this technology to a planetary exploration context is a non-trivial problem. A planetary surface is a harsh environment that differs from the structured environments found on Earth, and the platform would lack the support of a Global Navigation Satellite System (GNSS) to support localisation. Furthermore the limited computational resources, limited power and restricted choice of sensors means the large number of features commonly used for SLAM are not well suited to planetary hardware.

The PM-SLAM system developed in this thesis (Section 3) is concerned with developing a SLAM system suitable for planetary rovers using monocular cameras. The use of monocular imagery is motivated by the desire to simplify the hardware and software requirements of the system, as well as to develop a system that could be implemented on micro rover platforms.

Being able to localise a planetary rover within a global absolute map also has benefits, especially where the rover will be completing long traverses with specific predefined targets, such as in an SFR mission. Maps generated by extracting features from or-
bital imagery provide data that could enable a rover to localise absolutely within an
independent map. This helps the rover provide accuracy in terms of reaching its destin-
atation, and can also be useful in multi-rover scenarios such as the Forward Acquisition
of Soil and Terrain data for Exploration Rover (FASTER) mission scenario. Multiple
rovers localising within the same global map and sharing feature data would be able
to localise each other and add higher resolution data to each other’s maps. A proof
of concept system design is investigated, that fuses locally observed features with the
orbiter derived map to aid in localisation (Section 4).

1.3 Contribution

The contributions of this thesis in the area of autonomous rover localisation for plan-
etary exploration is summarised as follows:

1. Development of a SLAM system for use in planetary exploration:

The PM-SLAM system is a modular implementation of SLAM techniques for localising
a planetary vehicle using monocular imagery combined with odometry data.

Hybrid-salient features have been developed in order to achieve the goal of robustly
tracking sparse homogeneous features accurately, whilst maintaining efficiency in terms
of SLAM filter computation. The accurate tracking capability of point based feature
techniques has been combined together with more semantically representative salient
features.

The Hybrid-saliency algorithm is evaluated using the MOTA metric. A data set of im-
ages collected in a landscape representative for planetary exploration has been manually
annotated for use in evaluating the tracking accuracy.

A geometric method for calculating depth in monocular image is extended to determine
the position of a feature on the ground plane, in the local frame of the rover. An analysis
has been performed to determine the limitations of using this method in localisation
applications.

One simulated dataset, generated using PANGU and a two real-life datasets collected
1.4 Thesis Structure

The thesis is structured as follows:

- **Literature Survey:** Chapter 2 introduces a number of key concepts important for the research work, i.e., methods for autonomous localisation in terrestrial scenarios, the state of the art localisation methods used for current planetary exploration missions, SLAM systems and estimation filters used for localisation error reduction, the types of sensor data and feature properties used to help localise robotic platforms and the use of orbital imagery in planetary exploration missions.
• **A SLAM based planetary localisation system using monocular images**
  Chapter 3 presents Planetary Monocular SLAM, a autonomous localisation framework designed for planetary environments. Several filters are implemented in a modular system, and hybrid-salient features are presented as a means of making a suitable and robust input to the localisation system using monocular images from a planetary rover. The hybrid-saliency system itself is constructed in a modular manner allowing for configuration and improvement. The system is able to localise more accurately than a commonly used state-of-the-art Visual Odometry method and has a low computational overhead.

• **Use of orbital imagery for localising a planetary rover in a global map:**
  Chapter 4 presents a method of using orbital imagery in order to localise a rover within a global map, as opposed to a local relative map. The system makes use of the hybrid-salient techniques and the monocular depth perception algorithm introduced in Chapter 3 together with a global map created using pre-processed data from orbiters.

• **Summary and Future Work:** Chapter 5 presents conclusions drawn from the research results, along with potential directions for future research in this area.

A description of the datasets used in this thesis and some additional experimental results can be found in the appendices.
2.1 Planetary Localisation

Localisation of rovers on a planetary surface is a considerably challenging problem. The surface is likely to be sandy, meaning that reliance on dead-reckoning from systems such as wheel odometry would be likely to give poor location estimates due to the mobility of the terrain and the likelihood of wheel-slip. In addition, the surface environment is likely to be homogeneous with a low number of landmarks, sparsely distributed which places a large burden on any feature detection algorithm used as part of a visual localisation system.

Further complications arise from the nature of the limited hardware available for planetary missions. The computational power of space qualified hardware is much smaller than that available to terrestrial rovers. For example NASA’s Mars Science Laboratory (MSL) has a 200MHz processor, with 256MB of RAM \cite{10}. The on-board computer is shared amongst all of the rover’s tasks, including science tasks, communications as well as the GNC elements of the rover operations. Therefore the available computation is further limited.
Rovers are also restricted in the sensors available for navigation tasks. The use of LIDAR in space poses technical problems, such as the large energy requirement, the moving parts involved in scanning and the performance of LIDAR devices in dusty environments. A Compact Fast Scanning LIDAR (CFSL) has been developed with the goal of meeting these requirements and becoming space qualified \[11\], and has been tested by NASA at a terrestrial test facility \[78\], but to date no planetary surface exploration missions have been flown with a LIDAR on-board.

The NASA MER and MSL rovers utilised a GNC stack named GESTALT \[14\]. GESTALT operates an Sense-Model-Plan-Act (SMPA) architecture, which generates a detailed local hazard map using stereo-pairs in order to assess the safety of the local terrain. These maps take 3 minutes to generate and are created every 1.5m - 2m of traverse, however this information is not used currently for localisation and is discarded once the hazards have been passed. A localisation estimate is performed 8 times per second based on wheel odometry and IMU measurements (for heading determination). When operating in autonomous drive mode, the MER rovers are restricted to specific traverse segments, consisting of straight lines, arcs, and spot turns \[58\]. The localisation estimate is assisted by Visual Odometry (VO), which is performed during autonomous drives. In order to operate the VO system requires a 60\% overlap between successive images, which limits the motion between VO updates to 75cm movement and a maximum of 18° change in heading. Using the VO system with the on-board 20MHz RAD6000 processor requires 2 minutes per step. Therefore the use of VO to support localisation in autonomous drive mode is limited to areas where significant slippage is expected. MSL implements a similar approach to autonomous drives, but the increased power allows these modes to be used more often. Table 2.1 gives a summary of the autonomous drive modes available to the MSL, along with the rate of traverse achieved by each of them.

The Exomars GNC system has been designed to allow for a 70 metre/sol autonomous traverse, with no intervention from the ground \[95\]. It enables the rover to move to within 7 metres and ±5° of its selected target over the course of 2 sols. This is performed using Visual Localisation (VL), a VO algorithm which attempts to localise the rover once every 10 seconds \[72\]. In between the VL estimates, the rover is localised
2.1. Planetary Localisation

<table>
<thead>
<tr>
<th>Drive mode</th>
<th>Usage</th>
<th>Drive rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blind</td>
<td>No obstacles, low-slope terrain</td>
<td>3.17 cm/s</td>
</tr>
<tr>
<td>AutoNav</td>
<td>Obstacles, rugged terrain, scarps</td>
<td>1.25 cm/s</td>
</tr>
<tr>
<td>Visodom</td>
<td>High-slope terrain, cohesionless terrain</td>
<td>0.81 cm/s</td>
</tr>
<tr>
<td>AutoNav + Visodom</td>
<td>Rugged, sloped terrain</td>
<td>0.56 cm/s</td>
</tr>
</tbody>
</table>

Table 2.1: MSL rover drive modes, usage and estimated drive rates [35].

using IMU and odometry data. The computation architecture of the Exomars rover includes a main CPU and a co-CPU, which will both be 100MHz LEON2 processors. The co-processor will be dedicated to image processing tasks, rather than being shared amongst the other rover operational tasks. This means that, using VL in autonomous drive mode, the rover is able to move continuously.

The use of SLAM for planetary localisation has been investigated for hypothetical future missions, including the scenario of automated vehicles for planetary work sites [102]. In this scenario the rover would be restricted to a limited area, and loop closure would occur frequently. Exploration and sample return missions would involve longer trajectories with few opportunities for loop closure. The scale of the maps would also be larger for exploration missions. Some initial work has been done investigating the use of SLAM for longer planetary traverses [90]. Implementing SLAM for planetary exploration is not trivial, as the environment and platform present significant challenges. Terrestrial SLAM techniques have tended to be used either indoors [93], where the structured environment makes feature tracking a simpler task, or when used outdoors, SLAM systems have tended to use Global Positioning Data (GPS), and structured urban environments [87]. Planetary exploration provides a much more challenging scenario, as the terrain is unstructured and fairly homogeneous and GPS data is unavailable.
2.2 Simultaneous Localisation and Mapping

2.2.1 Overview of SLAM

Simultaneous Localisation And Mapping (SLAM) is a probabilistic technique for estimating the position of a mobile agent in an unknown environment while concurrently creating a map of local features. This is achieved using a combination of control data (the instructions used to make the rover move) and the model of the vehicle dynamics together with environmental observations. Both the control data and the environmental observations are assumed to be inherently noisy. A margin of error exists in the accuracy of the control data due to factors such as wheel-slip and the accuracy of the environmental observations will be dependent on the specification of the sensors used. By combining the information from both these inputs, their associated errors can be minimised [28], [100].

The main inputs to a typical SLAM system include a control signal, feature detection & tracking and a SLAM filter. The control signal and feature detection & tracking provide inputs to the SLAM filter, which is then able to estimate the overall rover state. Figure 2.1 shows how these inputs are used in each of the functional blocks SLAM, in order to predict, update and augment the localisation and map estimates.
2.2. Control Signal & Vehicle Model

The control signal is usually either the control commands sent to the rover to instruct it to move, or in many cases the wheel odometry. For a wheeled robot this can take the form of instructions to the drive motors and steering servos or feedback from sensors on the wheels themselves. In order to be useful to the filter this information is simplified into the incremental distance traversed \( (d) \) and the change in the heading angle \( (\phi) \) as shown in Equation 2.1. For the purposes of being concise these two parameters will be written as \( u_{d,t} \) and \( u_{\phi,t} \) at the time, \( t \), of each SLAM predict iteration. The change in heading angle \( u_{\phi,t} \) represents the anticlockwise deflection of the rover’s heading from its heading determined in the previous SLAM update step \( r_{\theta,t-1} \).

\[
\begin{bmatrix}
    d \\
    \phi
\end{bmatrix}
\]  
(2.1)

The vehicle model relates the rover’s new position \( (x_t) \) to its old position \( (x_{t-1}) \) based on the control command (Equation 2.1). In the two-dimensional case the rover position is represented by the \( x - y \) co-ordinates of the rover \( (r_{x,t} \text{ and } r_{y,t}) \) and the rover heading \( (r_{\theta,t}) \). These cartesian co-ordinates are given with respect to the rover’s position at the start of the traverse, i.e. at SLAM iteration \( t = 0 \). Therefore the origin of the rover map is \( (r_{x,0}, r_{y,0}) \) and the the rover headings are measured as the anticlockwise deflection from the initial heading, \( r_{\theta,0} \). The new rover position is calculated using a motion prediction function \( (f) \) as shown in Equation 2.2.

\[
\begin{bmatrix}
    r_{x,t} \\
    r_{y,t} \\
    r_{\theta,t}
\end{bmatrix} =
\begin{bmatrix}
    r_{x,t-1} \\
    r_{y,t-1} \\
    r_{\theta,t-1}
\end{bmatrix} +
\begin{bmatrix}
    -u_{d,t} \sin r_{\theta,t-1} \\
    u_{d,t} \cos r_{\theta,t-1} \\
    u_{\phi,t}
\end{bmatrix}
\]  
(2.2)

2.2.3 Feature Detection & Measurement Model

As well as using knowledge of the vehicle’s planned path, a SLAM system uses observations from the environment to improve its location estimate. In order to make use
Figure 2.2: Movement of the rover having been given a control vector. The axes in the bottom left corner represent the origin of the map and all co-ordinates are relative to it. The red circle represents the rover at time $t-1$ before the control signal and the blue circle represents rover at time $t$. The solid lines in the circles represent the heading angles.

of these observations features need to be extracted from the sensor data and stored so that they can be recognised and tracked if re-observed after the platform has moved. In terrestrial SLAM implementations LIDAR points are commonly used as features, however to date no LIDAR has been flown on a planetary surface mission, and the performance of such a device in the harsh conditions, considering the sparse terrain, dust levels, moving parts and power consumption is not well understood. Most planetary exploration rovers, however, have cameras, and a monocular camera is the simplest and richest form of perception for a rover vision system. Therefore a vision based SLAM that depends on monocular camera images is well suited to planetary applications. In monocular SLAM, environmental observations are in the form of images taken by a
single camera mounted on a rover. The different methods of feature extraction are described in Chapter 2.3.

The 2-D image plane locations of the detected features are translated into the real-world 3-D co-ordinates of the rover’s local frame in order to build a map of the environment and to be able to use the features to localise the rover. These local co-ordinates can be easily converted into a range \( z_r \) and a bearing \( z_\phi \) as shown in Equation 2.3, much like LIDAR data commonly used in SLAM systems.

\[
\begin{align*}
    z_r &= \sqrt{x^2 + y^2} \\
    z_\phi &= \tan \frac{x}{y}
\end{align*}
\]  

The range and bearing of a detected feature in the rover’s local frame can then be related to the position of the object in the global map \( (m_x, m_y) \) using the measurement model in Equation 2.4. The relationship between the rover position, the measurement and the feature position are shown in Figure 2.3.

\[
m = h(r, z) = \begin{bmatrix} m_x \\ m_y \end{bmatrix} = \begin{bmatrix} r_x \\ r_y \end{bmatrix} + \begin{bmatrix} -z_r \sin(r_\theta + z_\phi) \\ z_r \cos(r_\theta + z_\phi) \end{bmatrix}
\]  

2.2.4 SLAM Filters

The SLAM filter is the heart of a SLAM system. The rover’s knowledge of its current position and its local map (the rover state, \( x \)) is estimated from a probability distribution, conditioned upon the state at all previous times, and all the previous control data \( (u) \) and measurements \( (z) \) \[28\].

\[
P(x_t | x_{0:t-1}, z_{0:t-1}, u_{0:t})
\]  

A Markov assumption is made in SLAM, meaning that the current state of the rover is only dependent on the current inputs, and all knowledge of past inputs is contained
Figure 2.3: Measurement model. The red circle represents the rover, and the red line represents the heading angle. The axes in the bottom left corner represent the origin of the map and all co-ordinates are relative to them.
within the belief state from the previous step, i.e. the state is complete. The algorithm is therefore recursive and the state probability distribution can be written as:

$$P(x_t|x_{t-1}, u_t)$$

(2.6)

This probability is called the state transition probability. We can also make the assumption that if the state is complete, the probability of observing features in the environment can be defined as:

$$P(z_t|x_0:t, z_0:t, u_0:t) = P(z_t, x_t)$$

(2.7)

This distribution is defined as the measurement probability and allows us to predict the measurements given the complete state of the rover. These two probabilities allow us to generate a belief state for the rover state.

The belief state is the robots posterior estimate of its current state. Given the control signal inputs $u_0:t$ and the environmental observations $z_0:t$ for each step, the current state $x_t$ is represented by the belief function:

$$bel(x_t) = P(x_t|z_0:t, u_0:t)$$

(2.8)

The belief state is built using three main steps, a predict step, an update step and an augment step. These steps are illustrated in Figure 2.4.

In Figure 2.4a the rover has previously observed two features, the two black stars. Two further features have not yet been detected, shown as the two white stars. Using the control signal described in Equation 2.1, the hypothesised state is moved distance $u_{t,d}$ and turns $u_{t,\phi}$. This prediction is the a-priori belief $bel(x_t) = P(x_t|x_{t-1}, u_t)$. The true position of the rover, having moved, is shown in Figure 2.4b. The rover’s true state, shown in red, differs from the a-priori predicted state due to physical factors of the terrain and the vehicle itself. In Figure 2.4c the rover uses the a-priori state to predict the position of previously detected features (the red stars). Sensors then make measurements of the features’ actual locations (the black stars) with respect to
Figure 2.4: Stages of full SLAM iteration. (a) Rover moves forward from bottom of the figure. Two previously encountered features are shown as black stars. The white stars are newly observed features. (b) SLAM predict step produces an estimate of the rover position in red. (c) Rover uses estimated position to predict the position of previously observed features. These predictions are shown as red stars. (d) SLAM update step uses re-observed features to improve rover position estimate, shown in blue. (e) Using the improved estimate the SLAM augment step adds newly observed features to its map.
2.2. Simultaneous Localisation and Mapping

the rover’s true position. The difference between the predicted and measured feature positions (called the innovation) is used to update the rover’s belief state giving a posterior estimate of the rover position, shown in blue in Figure 2.4d, which is closer to the true position. Finally any newly observed features are added to the rover’s belief state, augmenting the map and providing reference points for future update steps (Figure 2.4e).

There are several existing SLAM filter designs, including the Gaussian filters such as the Extended Kalman Filter (EKF) and the Extended Information Filter (EIF), as well as Monte Carlo based particle filters such as FastSLAM. These are described in the following sections.

2.2.4.1 Extended Kalman Filter

The Kalman filter is a simple Bayesian estimation technique. It is commonly used for estimating the state of linear Gaussian systems [49]-[69]. The belief function is a multivariate Gaussian distribution, which is parameterised as a mean \( \mu_t \) and a covariance \( \Sigma_t \) for any given time \( t \).

\[
P(x) = \text{det}(2\pi\Sigma)^{-\frac{1}{2}} \cdot \exp \left\{ -\frac{1}{2}(x - \mu)^T\Sigma^{-1}(x - \mu) \right\}
\]  

(2.9)

Both the state transition probability and the measurement probability are assumed to be linear functions with zero-mean Gaussian noise. The predict, update and augment stages produce a mean state vector and a covariance matrix representing the Gaussian distribution of the belief state.

The vehicle model (Equation 2.2) and the measurement model (Equation 2.4) are both non-linear, which makes the Kalman filter unsuitable for estimating the rover state. Therefore the extended Kalman filter is used. The vehicle model and measurement model are both linearised, using a first order Taylor series approximation.

\[
r_t = f(r_{t-1}, u_t) \approx f(\mu_{t-1}, u_t) + f'(\mu_{t-1}, u_t)(r_{t-1} - \mu_{t-1})
\]

\[
= f(\mu_{t-1}, u_t) + F_t(r_{t-1} - \mu_{t-1})
\]

(2.10)
\[ m = h(r_t, z_t) \approx h(\mu_t, z_t) + h'(\mu_t, z_t)(r_t - \mu_t) \]
\[ = h(\mu_t, z_t) + H(r_t - \mu_t) \quad (2.11) \]

The predict step of the EKF produces an a-priori estimate \( \mu_t \) using the vehicle model (Equation 2.2) with the previous mean \( \mu_{t-1} \) as the previous state input.

\[ \bar{\mu}_t = f(\mu_{t-1}, u_t) \quad (2.12) \]

The a-priori covariance matrix is updated using the linearised motion model (Equation 2.10) together with the previous covariance \( \Sigma_{t-1} \) added to the control noise covariance.

\[ \bar{\Sigma}_t = F_t \Sigma_{t-1} F_t^T + R_t \quad (2.13) \]

In order to incorporate the information from the sensor measurements to produce a posterior mean and covariance the Kalman gain \( K_t \) is calculated. The Kalman gain represents the weight to be given to the measurement information. It is calculated using the a-priori covariance (Equation 2.13), the linearised measurement model (Equation 2.11) and the measurement noise covariance.

\[ K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t^{-1})^{-1} \quad (2.14) \]

The posterior mean is updated using the innovation (the difference between the predicted observation and the actual observation) and the posterior covariance is updated using the Kalman gain and the a-priori covariance.

\[ \mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t)) \]
\[ \Sigma_t = (I - K_t H_t)\bar{\Sigma}_t \quad (2.15) \]

Calculation of the covariance involves matrix multiplication and addition for prediction and matrix inversion, multiplication and addition in the update step.
2.2. Simultaneous Localisation and Mapping

<table>
<thead>
<tr>
<th></th>
<th>Calculate a-priori mean.</th>
<th>$\bar{\mu}<em>t = f(\mu</em>{t-1}, u_t)$</th>
<th>Predict</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.</td>
<td>Calculate a-priori covariance.</td>
<td>$\bar{\Sigma}<em>t = F_t\Sigma</em>{t-1}F_t^T + R_t$</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>Calculate Kalman gain.</td>
<td>$K_t = \bar{\Sigma}_t H_t^T (H_t\bar{\Sigma}_t H_t^T + Q_t^{-1})^{-1}$</td>
<td>Update</td>
</tr>
<tr>
<td>4.</td>
<td>Make measurements.</td>
<td>$z_t$</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>Calculate posterior mean.</td>
<td>$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>Calculate posterior covariance.</td>
<td>$\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>Add new features to map.</td>
<td>Augment</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2: Steps of an Extended Kalman Filter SLAM system.

Each feature that is detected grows the state vector and the covariance matrix. The calculation of the Kalman gain requires a matrix inversion, which is a complex operation for large matrices. This means that the update steps take time quadratic in the number of landmarks in the map.

2.2.4.2 Extended Information Filter

The information filter is a dual of the Kalman filter. Instead of parameterising the Gaussian distribution using the mean and the covariance, the it uses the information vector $\eta$ and the information matrix $\Lambda$. These parameters are derived by rearranging the multivariate Gaussian probability density function in Equation 2.13 [101].

\[
\begin{align*}
P(x) &= \det(2\pi\Sigma)^{-\frac{1}{2}} \cdot \exp \left\{ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right\} \\
&= \alpha \cdot \exp \left\{ -\frac{1}{2} x^T \Sigma^{-1} x + x^T \Sigma^{-1} \mu \right\} \\
&= \alpha \cdot \left\{ -\frac{1}{2} x^T \Lambda x + x^T \eta \right\} \\
\Lambda &= \Sigma^{-1} \\
\eta &= \Sigma^{-1} \mu = \Lambda \mu
\end{align*}
\]

The Extended Information Filter (EIF), like the EKF, requires the vehicle model and measurement model to be linearised and modelled as Gaussian as in Equation 2.10 and Equation 2.11. The predict step of the EIF produces an a-priori information vector.
and information matrix using the vehicle model Equation 2.2 with the mean recovered from the previous step $\mu_{t-1}$ as the previous state input.

$$
\bar{\Lambda}_t = (F_t \Lambda_{t-1}^{-1} F_t^T + R_t)^{-1}
$$

(2.17)

$$
\mu_{t-1} = \Lambda_{t-1}^{-1} \eta_{t-1}
$$

$$
\bar{\eta}_t = \bar{\Lambda}_t f(\mu_{t-1}, u_t)
$$

(2.18)

The posterior information matrix is updated by adding the a-priori information matrix to the inverse of the measurement noise. The information vector is updated using the innovation (the difference between the predicted observation and the actual observation) and the a-priori mean.

$$
\Lambda_t = \bar{\Lambda}_t + H_t^T Q_t^{-1} H_t
$$

(2.19)

$$
\bar{\mu}_t = f(\mu_{t-1}, u_t)
$$

$$
\eta_t = \bar{\eta}_t + H_t^T Q_t^{-1} [z_t - h(\bar{\mu}_t) + H_t \bar{\mu}_t]
$$

(2.20)

The predict step of this algorithm involves a matrix inversion, and the update step involves matrix addition, the converse of the EKF (see chapter 2.2.4.1). In order to take advantage of this the information matrix needs to be kept sparse, which reduces the complexity of the steps. This is achieved by marginalising out features from the posterior information matrix. This is equivalent to breaking the links between features which are no longer visible to the rover. This step is called sparsification. The features are divided into three subsets: features previously visible to the rover that are still visible ($m_+$), features that were previously visible but are no longer visible ($m_0$) and the set of features that were not visible in this or the previous step ($m_-$. The posterior
2.2. Simultaneous Localisation and Mapping

distribution can then be written in terms of the rover pose $S_t$ and these feature subsets.

\[
x_t = \langle S_t, m \rangle \\
m = m_+ + m_0 + m_- \tag{2.21}
\]

\[
P(x_t|z_{0:t}, u_{0:t}) = P(S_t, m_+, m_0, m_-|z_{0:t}, u_t)
\]

The rover pose is not dependent on the passive features therefore this parameter is set to an arbitrary value. The set of features $m_0$ can then be marginalised out from the posterior using an approximation \[31\].

\[
\bar{P}(S_t, m|z_{0:t}, u_{0:t}) \\
\frac{P(S_t, m_+, m_0, m_-|z_{0:t}, u_{0:t})}{P(m_+, m_0, m_-|z_{0:t}, u_{0:t})} P(m_- = 0, z_{0:t}, u_{0:t}) \tag{2.22}
\]

The sparsification and the additive update step mean that this algorithm has a constant update time \[105\].

### 2.2.4.3 FastSLAM

The FastSLAM algorithm is based on particle filters \[74\], \[84\]. Particle filters are a type of non-parametric filtering technique that model the posterior through a set of samples $\chi$ randomly drawn from the posterior. These random samples are called particles.

\[
\chi_t = x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]} \\
x_t^{[m]} \sim P(x_t|z_{0:t}, u_{0:t}) \tag{2.23}
\]

It would not be practical to estimate the large number of variables in SLAM using a particle filter. FastSLAM is an algorithm that takes advantage of the fact that features in the map are conditionally independent of each other. Therefore we can factor the posterior distribution into the distribution of the rover trajectory $S_{0:t}$ and the independent distributions of each map feature $m_n$.

\[
P(x_{0:t}|z_{0:t}, u_{0:t}) = P(S_{0:t}|z_{0:t}, u_{0:t}) \prod P(m_n|x_{0:t}, z_{0:t}) \tag{2.24}
\]
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Recover previous mean.</td>
<td>( \mu_{t-1} = \Lambda_{t-1}^{-1}\eta_{t-1} )</td>
</tr>
<tr>
<td>2</td>
<td>Calculate a-priori information matrix.</td>
<td>( \hat{\Lambda}<em>t = (F_t\Lambda</em>{t-1}^{-1}F_t^T + R_t)^{-1} )</td>
</tr>
<tr>
<td>3</td>
<td>Calculate a-priori information vector.</td>
<td>( \bar{\eta}_t = \hat{\Lambda}<em>tf(\mu</em>{t-1},u_t) )</td>
</tr>
<tr>
<td>4</td>
<td>Make measurements.</td>
<td>( z_t )</td>
</tr>
<tr>
<td>5</td>
<td>Calculate posterior information vector.</td>
<td>( \eta_t = \bar{\eta}_t + H_t^TQ_t^{-1}[z_t - h(\bar{\mu}_t) + H_t\bar{\mu}_t] )</td>
</tr>
<tr>
<td>6</td>
<td>Calculate posterior information matrix.</td>
<td>( \Lambda_t = \bar{\Lambda}_t + H_t^TQ_t^{-1}H_t )</td>
</tr>
<tr>
<td>7</td>
<td>Add new features to map.</td>
<td>Augment</td>
</tr>
<tr>
<td>8</td>
<td>Sparsify information matrix.</td>
<td>( \hat{\Lambda}<em>t = \Lambda_t - \Lambda_0^0F</em>{m0}(F_{m0}^T\Lambda_0^0F_{m0})^{-1}F_{m0}^T\Lambda_0^0 + \Lambda_0^0F_{x,m0}(F_{x,m0}^T\Lambda_0^0F_{x,m0})^{-1}F_{x,m0}^T\Lambda_0^0 - \Lambda_0^x(F_x^T\Lambda_0^0F_x)^{-1}F_x^T\Lambda_t )</td>
</tr>
<tr>
<td>9</td>
<td>Sparsify information vector.</td>
<td>( \bar{\eta}_t = \eta_t + \mu_t(\bar{\Lambda}_t - \Lambda_t) )</td>
</tr>
</tbody>
</table>

**Table 2.3:** Steps of an Extended Information Filter SLAM system.
FastSLAM uses a version of the particle filter called a Rao-Blackwellised particle filter. The posterior over the rover pose is estimated using particle filters, and the map features are each represented by a Gaussian filter like those in Chapters 2.2.4.1 and 2.2.4.2. Each particle therefore contains a sample of the rover pose and N sets of Gaussian parameters, one for each feature in the map.

The predict step is carried out by sampling the rover pose from the previous posterior.

\[
x_t^{[k]} \sim P \left( x_t | x_{t-1}^{[k]}, u_t \right) \tag{2.25}
\]

For each particle k, EKFs for the new features are added and the EKFs for the re-observed features are updated using Equations 2.14 and 2.15. Once all the EKFs have been updated an importance weight is calculated.

\[
w^{[k]} = |2\pi Q|^{-\frac{1}{2}} \cdot exp \left\{ -\frac{1}{2} (z_t - \tilde{z}_n)^T Q^{-1} (z_t - \tilde{z}_n) \right\} \tag{2.26}
\]

These importance weights are then used to proportionately resample the particles.

### 2.2.4.4 Summary of SLAM Filters

For a summary of the SLAM filters discussed in the previous chapters, see Table 2.5.

<table>
<thead>
<tr>
<th>Particle</th>
<th>Robot path</th>
<th>Feature 1</th>
<th>Feature 2</th>
<th>Feature N</th>
</tr>
</thead>
<tbody>
<tr>
<td>k=1</td>
<td>(x_{0:t}^{[1]})</td>
<td>(\mu_1^{[1]}, \Sigma_1^{[1]})</td>
<td>(\mu_2^{[1]}, \Sigma_2^{[1]})</td>
<td>...</td>
</tr>
<tr>
<td>k=2</td>
<td>(x_{0:t}^{[2]})</td>
<td>(\mu_1^{[2]}, \Sigma_1^{[2]})</td>
<td>(\mu_2^{[2]}, \Sigma_2^{[2]})</td>
<td>...</td>
</tr>
<tr>
<td>k=M</td>
<td>(x_{0:t}^{[M]})</td>
<td>(\mu_1^{[M]}, \Sigma_1^{[M]})</td>
<td>(\mu_2^{[M]}, \Sigma_2^{[M]})</td>
<td>...</td>
</tr>
</tbody>
</table>

---

**Figure 2.5:** Structure of FastSLAM particles [100].
Chapter 2. Literature Survey

1. For each particle:

   a. Sample a pose from the previous belief state.

   \[ x_t^{[k]} \sim P \left( x_t | x_{t-1}^{[k]}, u_t \right) \]

   Predict

   b. Initialise mean and covariance for new features with default importance weight.

   \[
   \begin{align*}
   \mu_{t,n}^{[k]} &= h^{-1}(z_t, x_t^{[k]}) \\
   \Sigma_{t,n}^{[k]} &= H^{-1}Q_t(H^{-1})^T
   \end{align*}
   \]

   Augment

   c. For the existing features update mean and covariance using EKF and calculate importance weight.

   \[
   \begin{align*}
   \mu_{t,n}^{[k]} &= \mu_{t-1,n}^{[k]} + K(z_t - \bar{z}) \\
   \Sigma_{t,n}^{[k]} &= (I - KH)\Sigma_{t-1,n}^{[k]}
   \end{align*}
   \]

   Update

4. Resample particles using importance weights.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>For each particle:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a.</td>
<td>Sample a pose from the previous belief state.</td>
<td>[ x_t^{[k]} \sim P \left( x_t</td>
<td>x_{t-1}^{[k]}, u_t \right) ]</td>
</tr>
</tbody>
</table>
| b. | Initialise mean and covariance for new features with default importance weight. | \[
   \begin{align*}
   \mu_{t,n}^{[k]} &= h^{-1}(z_t, x_t^{[k]}) \\
   \Sigma_{t,n}^{[k]} &= H^{-1}Q_t(H^{-1})^T
   \end{align*}
   \] | Augment |
| c. | For the existing features update mean and covariance using EKF and calculate importance weight. | \[
   \begin{align*}
   \mu_{t,n}^{[k]} &= \mu_{t-1,n}^{[k]} + K(z_t - \bar{z}) \\
   \Sigma_{t,n}^{[k]} &= (I - KH)\Sigma_{t-1,n}^{[k]}
   \end{align*}
   \] | Update |
| 4. | Resample particles using importance weights. |   |   |

**Table 2.4:** Steps of a FastSLAM system.
### 2.2. Simultaneous Localisation and Mapping

#### Parameterisation of $P(x_t|z_{0:t}, u_{0:t})$

<table>
<thead>
<tr>
<th></th>
<th>EKF</th>
<th>EIF</th>
<th>FastSLAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(x_t</td>
<td>z_{0:t}, u_{0:t})$</td>
<td>$exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$</td>
<td>$exp(-\frac{1}{2}x^T\Lambda x+\eta x)$</td>
</tr>
</tbody>
</table>

#### Key Parameters

<table>
<thead>
<tr>
<th></th>
<th>Covariance matrix: $\Sigma$</th>
<th>Information matrix: $\Lambda = \Sigma^{-1}$</th>
<th>$N$ particles for path estimate</th>
<th>$N \cdot M$ feature EKFs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean vector: $\mu$</td>
<td>Information matrix: $\eta = \mu \Sigma^{-1}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Prediction step

<table>
<thead>
<tr>
<th></th>
<th>Additive</th>
<th>Matrix inversion</th>
<th>Additive</th>
</tr>
</thead>
</table>

#### Update step

<table>
<thead>
<tr>
<th></th>
<th>Matrix inversion</th>
<th>Additive</th>
<th>Matrix inversion</th>
</tr>
</thead>
</table>

#### Advantages

<table>
<thead>
<tr>
<th></th>
<th>• Accurate and simple estimation model</th>
<th>• Sparse information matrix reduces computation</th>
<th>• Optimal estimator</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>• Update step is additive further reducing computation</td>
<td>• Does not rely on linear model</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Does not assume Gaussian distribution</td>
</tr>
</tbody>
</table>

#### Disadvantages

<table>
<thead>
<tr>
<th></th>
<th>• Update step uses matrix inversion - computationally expensive</th>
<th>• Non optimal estimator</th>
<th>• Does not always converge</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Complexity increases quadratically with map size</td>
<td>• Linearised models used</td>
<td>• Accuracy dependent on number of particles - trade off with computational expense</td>
</tr>
<tr>
<td></td>
<td>• Non optimal estimator</td>
<td>• Gaussian distribution used</td>
<td>• Non optimal estimator</td>
</tr>
<tr>
<td></td>
<td>• Linearised models used</td>
<td></td>
<td>• Linearised models used</td>
</tr>
<tr>
<td></td>
<td>• Gaussian distribution used</td>
<td></td>
<td>• Gaussian distribution used</td>
</tr>
</tbody>
</table>

| Table 2.5: Comparison of common SLAM filters [90]. |
2.3 Feature Detection

2.3.1 Point Based Feature Detection

Point based feature detection refers to techniques for extracting local interest points in an image, by analysing the surrounding pixels. A descriptor provides information about the pixel and its context and can be used for tracking the feature across multiple images as well as for classifying the features. These class of features are commonly used in applications such as stereo image re-construction, for instance [75], where corners were detected in the image. Harris corners [43] are a firmly established technique, and have been used in motion tracking systems as well as 3D reconstruction [42]. The algorithm works by selecting pixels which have large gradients in all directions. In [88] extracted Harris corner features were stored in a database, with associated descriptors in order to perform tracking over multiple consecutive images. Harris corners are very sensitive to scale and orientation changes. This means that a moving camera which takes images of an object as it moves towards it, may not be able to track the object robustly using Harris features alone.

Several point based feature extraction and tracking schemes have been developed to address the problems of scale, illumination and orientation variance and several algorithms have been developed offering different levels of performance in terms of accuracy and computation.

2.3.1.1 Scale Invariant Feature Transform

Scale Invariant Feature Tracking (SIFT) is a commonly used image feature and extraction algorithm [63]. SIFT was designed to solve the problem of Harris corner-type detectors being sensitive to rotation and scale. SIFT keypoints and descriptors are scale and rotation invariant, allowing matching between keypoints regardless of any scaling or rotation between two images. Detection of keypoints is performed using a cascade filtering approach, which identifies candidate locations of interest which are then analysed in further detail. In order to maintain scale invariance the initial step
calculates differences of Gaussians across a number of scales. Local minima and maxima are detected by comparing each sampled pixel with its neighbours in the current image and the adjacent scale images. The sampling frequency used by the algorithm is selected in order to ensure robust, repeatable point detection.

Once keypoints have been identified within the image, descriptors need to be generated for each of them. SIFT descriptors are calculated using the orientation of the keypoint, which is defined by the gradient and orientation based on pixel differences. For each keypoint a 4x4 area is defined around the pixel in the image plane and the orientation vectors within each subregion are converted to a histogram with 8 bins representing 45 degrees each. The 16, 8-bin histograms are concatenated into a 128 member vector, representing the keypoint descriptor.

Keypoint descriptors are matched by computing the euclidean distance between the descriptors of two keypoints. Compared to the techniques discussed later on in this chapter the descriptor vector is large meaning that this comparison stage is slower than other available algorithms. SIFT is however a robust and simple algorithm, and has been demonstrated on an FPGA [89]. Further attempts to improve the efficiency of SIFT have included methods such as PCA-SIFT [50], which reduces the descriptor vector to 36 elements increasing the matching speed. However the modified descriptor has less distinctiveness and its generation is less computationally efficient.

### 2.3.1.2 Speeded-Up Robust Features

The Speeded Up Robust Features (SURF) algorithm was developed to improve the performance of feature matching from SIFT in all three stages of the feature matching process [12]. One of the main advantages is the reduction in size of the descriptor, while maintaining a high distinctiveness. Like SIFT, SURF descriptors are designed to be scale and in-plane rotation invariant. The keypoint detector element of SURF is based on Hessian matrices. The detection is performed on integral images, where each pixel represents the summation of all the pixels within a rectangular area. The use of integral images allows the algorithm to function more efficiently. Blobs representing areas of interest are detected by computing the Hessian response of the integral image.
and selecting local maxima from the Hessian determinants. In order to detect scale invariant keypoints, detection is performed on several scales. Rather than scaling the image (as in SIFT), by using the integral image the filter can be scaled, thus removing issues with aliasing.

The process of generating a descriptor in SURF is similar to SIFT, in that it is derived from local intensity and gradient information. In SURF this information is extracted using a first order Haar wavelet response. The wavelet is applied to the integral image in a circular region around the keypoint. The responses are plotted in two dimensions and then summed within angular windows. The largest vector from these summed windows is used to represent the orientation of the point. A square area rotated to the orientation of and centred on the keypoint is defined and divided into 16 subregions. The Haar response of each subregion is sampled at 25 evenly spaced points, in both the x and y directions. All of the x responses and y responses in each subregion are summed to give the first two elements of each subregion descriptor vector. The absolute sums of these are also included to indicate the polarity of the subregion Haar response. These 4 elements are combined into a vector for each subregion, which are then concatenated
2.3. Feature Detection

Figure 2.7: An example of image feature detection and matching using the SURF algorithm. Two consecutive images are shown side-by-side from a simulated rover traverse. The red dots indicate detected SURF points, and the red lines represent the co-registration of points between the two images.

into a 64 element vector, which acts as a descriptor for the keypoint. Using the Haar wavelet response makes the resulting descriptor illumination invariant and less sensitive to noise than SIFT descriptors.

SURF descriptors are matched in a similar fashion to SIFT descriptors, by computing the euclidean distance between the two descriptors of two keypoints. The matching can be sped up by using the Laplacian, which is already calculated during the feature detection stage of the algorithm. The combination of the smaller descriptor and the ability to filter match candidates using the Laplacian mean that SURF matching is more computationally efficient than SIFT.

2.3.1.3 Binary Robust Invariant Scalable Keypoints

Binary Robust Invariant Scalable Keypoints (BRISK) is another keypoint detection and matching algorithm, designed to be computationally efficient, have robust, repeatable and distinctive descriptors and be able to match features at different scales and rotations [56]. BRISK differs from SURF and SIFT in both the feature extraction and descriptor calculation steps. The algorithm was developed by looking at combinations of keypoint
detectors and descriptors that offered benefits over SURF and SIFT. A combination of the FAST keypoint extractor for finding keypoints [85], together with descriptors BRIEF [17] have demonstrated to be suitable for real-time applications, however this combination does not offer robust rotation or scale invariance and as such would not be suitable in the context of SLAM on an outdoor platform [26].

The keypoint detection stage of BRISK is based on AGAST [64], which itself is an extension of the FAST algorithm, but with modifications to provide scale invariance. This is achieved by detecting maxima, not only in the image plane but also in scale-space, with the FAST score used as a measure of salience. The scale of the keypoint in BRISK is given as a continuous quantity. Features are detected using a FAST 9-16 detector, which analyses the brightness of 16 pixels surrounding the central pixel and requires 9 of these to be sufficiently different to meet the FAST criteria. This detector is applied across several scales using a constant threshold in order to identify local areas of interest.

As opposed to the vector of elements used in SURF and SIFT, the BRISK keypoint descriptor is in the form of a binary string, which is calculated from brightness comparisons of the keypoints and the surrounding pixels and attempts to maximise the distinctiveness. The first step of generating a descriptor is to determine the orientation of the keypoint, in order to ensure rotation invariance. This is determined by sampling points lying on a set of equally spaced concentric circles centred on the keypoint. The gradient between each of the sample points is calculated, and used to compute the overall characteristic pattern direction of the selected keypoint. The same sampling pattern used to determine the orientation of the keypoint, is rotated to align with the direction of the keypoint. A binary value of 1 or 0 is assigned to the intensity comparison between each pixel in sampling pattern. The complete descriptor is a 512 bit string calculated using this method.

BRISK keypoints can be matched in a similar way to other binary descriptors, for instance BRIEF. The Hamming distance between two keypoint descriptors is calculated as a measure of the dissimilarity. This is a very efficient method of comparing descriptors.
2.3. Feature Detection

Figure 2.8: An example of image feature detection and matching using the BRISK algorithm. Two consecutive images are shown side-by-side from a simulated rover traverse. The red dots indicate detected BRISK points, and the red lines represent the co-registration of points between the two images.

2.3.1.4 Outlier Rejection

When performing feature matching across frames, there will inevitably be some mismatches, or outliers. In the case where the feature matches are used as part of a vision-based localisation system these outliers tend to reduce the accuracy of the system. Therefore it is desirable to filter out these outliers before the feature set is used. There are a number of widely accepted methods that can provide a solution to this problem, such as using Random Sample Consensus (RANSAC) [32], or Rejection of Outliers by Rotations (ROR) [1]. Both these methods attempt to derive the rigid transformation between the matched pairs minimising the error for the most points. Those points whose errors exceed a threshold are counted as outliers and removed from the final list of matches. An example of this pruning is shown in Figure 2.9.

2.3.2 Salient Feature Detection

Visual saliency feature detection algorithms are biologically inspired methods for identifying Regions Of Interest (ROIs) in an image. These salient ROIs are based on image
Figure 2.9: (a) SURF pairs matched between two images before outlier rejection. Two consecutive images are shown side-by-side separated by a white line. Detected feature keypoints are shown as red circles, and correlated points between the two images are connected by red lines. (b) SURF pairs after outliers have been rejected using the RANSAC algorithm described in [32]. The number of correlations has been reduced by outlier rejection, but the remaining co-registered points are more accurate.
2.3. Feature Detection

characteristics such as colour, orientation or intensity across the image. The output of
the models are pseudo-semantic blobs, which represent key features in the image (e.g.
rocks on the planetary surface). These techniques can be applied to low-level object
detection and tracking \[59, 107, 108\], as well more complex localisation and navigation
applications \[94\].

Saliency models have been developed using both top-down and bottom-up approaches
\[15\]. Top-down algorithms require supervised learning techniques to give the system
information related to high level cognitive factors. Bottom-up models use low level
images, and do not require training in order to extract desired features of interest, and
are therefore more general and more suited to environments where a-priori knowledge
is limited.

2.3.2.1 Hou Saliency

The Hou saliency algorithm \[45\] is a computationally simple and fast algorithm for
detecting regions of interest within an image. The algorithm is designed to identify
unexpected elements in the visual scene by ignoring commonly occurring characteristics
and instead selecting areas that significantly differ from the rest of the image. The areas
of the image deemed statistically less relevant are then removed, leaving a heat map of
salient regions.

This analysis is performed in the frequency domain. The averaged spectrum is subtrac-
ted from the log spectrum, giving a result termed the spectral residual. This spectral
residual represents the parts of the image which differ significantly from the average
characteristic. When this residual is transformed back in to the spatial domain, using
an inverse Fourier transform the result is a saliency heat map which can then be
binarised.

2.3.2.2 Rudinac Saliency

The Rudinac saliency algorithm \[86\] provides an extension of the spectral residual tech-
nique employed in the Hou algorithm. While the Hou algorithm operates on grayscale
values, calculating the residual of the image intensity, Rudinac includes information based on colour-opponency analysis in order to determine salient regions in the image. Three channels are constructed from the RGB input:

- An intensity channel, representing the average of all three channels.
- A Red-Green difference (RG) channel, with normalised values.
- A Blue-Yellow difference (BY) channel, where the yellow channel is the lower of the red and green values. The Blue-yellow difference is also normalised.

The spectral residual for each of these three channels is calculated individually, producing three saliency maps. These saliency maps are weighted and combined to produce the final saliency map output. The weightings are determined by calculating local maxima across each of the saliency maps, and selecting the average peak values from each, similar to the method used in the Itti Koch Niebur saliency algorithm [47]. The resulting grayscale heatmap can then be binarised. In tests on images recorded at the Airbus Mars yard, the colour information helped to distinguish textures on sandy terrain from boulders in the visual scene.
2.3. Feature Detection

Figure 2.11: Output of salient blob detection using Rudinac algorithm on PANGU generated images. The left image is the original, and the right is the image with salient contours detected by the Rudinac algorithm outlined in red.

2.3.2.3 Thresholding Saliency Heatmaps

These algorithms output clusters of pixels using intensity-based weightings, essentially the ROIs present in the image. These blobs need to be converted into binary maps of salient features with hard outlines, separating them from the background, or non-salient elements of the image. The saliency map can be thresholded using the intensity values to achieve this. The blobs in the binarised images represent the location and borders of semantic elements in the visual scene of the input (e.g. rocks on the planetary surface). Otsu’s method [77] is commonly used for this conversion of a saliency map into a binary image made up of ROI blobs. The image is assumed to have a bimodal distribution, comprising salient pixels (those deemed to be within the ROIs) and the background (the remaining pixels). A histogram shape-based threshold is selected in order to binarise the saliency map. This method essentially follows an exhaustive search strategy to compute the optimum threshold that minimises the intra-class variance or maximises the inter-class variance in order to accurately determine the class for each pixel. The results of this process are shown in Figure 2.12.

2.3.2.4 Matching and Tracking of Salient Features

The salient feature extraction algorithms produce a contour for each detected landmark. Between consecutive images the outline contour of these features can change
Chapter 2. Literature Survey

Figure 2.12: (a) Grayscale heatmap produced by salient feature extraction. (b) Binarised blob mask produced from heatmap using Otsu’s method [77].

significantly. Therefore the contour information is not useful for being able to track the features across sequential images. One simple technique to enable tracking is using a k-nearest neighbour (k-NN) instance-based search algorithm [92]. The algorithm coregisters salient regions in subsequent frames by applying the Euclidean norm ($l^2$ norm) as a distance metric. Pairwise distances are measured between the centroids of salient ROI patches in the current and the previous image frames. Similar objects at each step are labelled as the same landmark, achieving a quasi-clustering of similar objects using a spatial and temporal similarity comparison, i.e. tracking of features.

The centroid of an ROI detected at time $t$ is used as a reference point $R^t_l$, where $l = \{0, 1, ..., n\}$ is the id used to label it. An exhaustive search for the centroid of the detected salient object is carried out in the previous frame ($t - 1$). This is the query point $Q^{t-1}_\gamma$ where $\gamma = \{0, 1, ..., n\}$ is the id such that the Euclidean distance between the two centroids is minimised. Once a pair of features have been identified they are given a common id $\mathcal{L} = \{0, 1, ..., n\}$ which is throughout subsequent frames in order to achieve tracking. This is formally described in Equation 2.27.

$$\forall l \forall \gamma, kNN(R^t_l) \rightarrow \mathcal{L}: \mathcal{L} = \arg\min_\gamma \{\|R^t_l - Q^{t-1}_\gamma\|\}$$ \hspace{1cm} (2.27)

An example of landmark matching using k-NN instance searching is shown in Figure 2.13. This type of matching works well when features are seen in sequential images,
2.4 Orbiter Image Localisation

Increasingly, orbital imaging platforms around Mars are capable of producing very high quality data. High resolution images of the Martian surface can be produced, thanks to new sensors combined with techniques such as super-resolution, the technique of combining imagery from several orbital passes and multiple platforms in order to increase the image quality beyond the baseline capability of the sensor itself. For instance the High Resolution Imaging Science Experiment (HiRISE), the camera installed on the Mars Reconnaissance Orbiter (MRO) launched in 2005, can achieve a resolution of 30cm per pixel [71].

Beyond their value as scientific data-products, orbital imagery has been used to support other Mars exploration activities. Several lander and rover missions to the Martian surface have utilised high resolution maps generated from orbiter imagery in order to select a suitable landing site [35,39,40,52]. Manual analysis has been used to identify the traversability and geological properties of candidate landing sites, and in order to define the safe landing ellipse to ensure the safety and success of the missions. Comparisons

---

**Figure 2.13:** Salient features in sequential images tracked using k-NN search. Detection is performed on the left image, and then on the right image after the camera has moved. The bounding boxes represent features, and the numbers represent the tracked id across the subsequent images.

However features that are re-observed later in the traverse cannot be re-identified unless the feature has been present in all the intermediate frames.
of the predicted terrain with the in-situ observations made after landing validated the use of orbital imagery for characterising the surface \[37, 38\]. The use of orbital imagery for aiding pinpoint landing on the Martian surface has also been suggested \[21\]. The method proposed compares a database of features derived from orbital image data, with images from the descent camera on the landing craft. Orbital images have also been used extensively for manually localising platforms during standard rover operations \[57, 80\]. Ground operators use the high resolution images together with images taken on the ground in order to determine the position of the rover on the planetary surface.

The successful use of orbital imagery throughout the history of Mars exploration missions suggests that this source of data could be used as part of a more autonomous rover GNC system. An investigation into next generation GNC architectures, based upon a model SFR mission included suggestions of new technologies that would be required for a successful mission with increased autonomous long-distance traverses \[91\]. These technologies included a localisation system termed Constellation Matching, based on the principle of autonomously matching features from orbiter images (the coarse map)
with features extracted from images captured by the rover’s on board camera (the local map), in order to aid with localising the rover.

In the proposed design groups of detected features in the local map (rocks) are coregistered with features visible in larger orbital maps, which will have been generated offline and preloaded onto the rover before launch. Previous rover missions to the Martian surface have required less than a year for the rover to land on the surface after launching from Earth [24] so any preloaded map would not be substantially out of date. Preloaded orbital maps have been considered of sufficient quality for use in pinpoint landings of craft on the planetary surface [21]. Furthermore the nature of the map, i.e. a database of cartesian co-ordinates of landmarks could easily be partially updated at any stage of the mission, without the need to transmit a new map of the entire traverse.

A terrestrial system used for identifying a rover in a dense forest using aerial and ground images serves as a starting point for the design [46] of a planetary Constellation Matching system. The centre of tree crowns are detected from aerial photographs taken by a flying platform. The ground segment takes LIDAR scans and detects the stems (trunks) of the surrounding trees. The system then attempts to correlate the arrangement of features detected on the ground with the features found in the aerial map. This is achieved using the Iterative Closest Point (ICP) algorithm. This algorithm iteratively minimises the error distance $E$ between all the matched points $p$ and $q$ (from the ground and the aerial data respectively), assuming that the maps are related by a rotation $R$ and a translation $t$ as shown in Equation 2.28.

\[
\arg\min_{p,q \in \mathbb{R}^2} E_{p,q} = \frac{1}{N} \sum_{i=1}^{N} \| \vec{R} \times \vec{p}_i + \vec{t} - \vec{q}_i \|^2 \tag{2.28}
\]

The system was tested using three aerial image sets at varying resolutions and a single LIDAR data set to represent the ground features (Table 2.6).

The algorithm performance was dependent on the aerial image resolution, with lower resolution images giving larger localisation errors (Table 2.7). The speculated cause of this correlation is the delineation of the dense tree crowns in the forest canopy, which is a scenario that would not apply to planetary localisation.
### Chapter 2. Literature Survey

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerial Set 1</td>
<td>5000x5000 pixels, 0.3m resolution</td>
</tr>
<tr>
<td>Aerial Set 2</td>
<td>5000x5000 pixels, 0.3m resolution</td>
</tr>
<tr>
<td>Aerial Set 3</td>
<td>8000x8000 pixels, 0.5m resolution</td>
</tr>
<tr>
<td>Ground Set (LIDAR)</td>
<td>5cm resolution</td>
</tr>
</tbody>
</table>

**Table 2.6:** Datasets used to evaluate ground and aerial feature matching for a autonomous terrestrial forest vehicle [46].

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Average Position Error (m)</th>
<th>Standard Deviation (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerial Set 1</td>
<td>5.41</td>
<td>4.42</td>
</tr>
<tr>
<td>Aerial Set 2</td>
<td>4.76</td>
<td>3.97</td>
</tr>
<tr>
<td>Aerial Set 3</td>
<td>6.83</td>
<td>5.63</td>
</tr>
</tbody>
</table>

**Table 2.7:** Results of rover localisation using coregistration of tree stems with tree crowns from ground and aerial platforms. The rover pose was estimated 234 times at intervals of 0.5m. [46].

An extra test was performed, which used manual classification of features in order to evaluate the map matching independently of the environment specific elements of the algorithm. The results were an improvement on the performance of the entire system (2.8).

In order to apply this technique to planetary exploration, the tree stem LIDAR data would be replaced with visual features extracted from images taken by the rover platform, and the map from the aerial images would be represented by features preloaded into the rover, derived from orbital imagery. Figure 2.15 describes how a planetary Constellation Matching based localisation system would aid in localising a rover. In Figure 2.15a the rover is given a planned path, shown in green. As the rover traverses

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Average Position Error (m)</th>
<th>Standard Deviation (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerial Set 2</td>
<td>2.79</td>
<td>3.38</td>
</tr>
</tbody>
</table>

**Table 2.8:** Results of rover localisation using coregistration of manually classified tree stems and tree crowns from ground and aerial platforms [46].
it attempts to follow this path, however due to a combination of wheel-slip and small
course corrections to avoid unexpected obstacles the rover actually follows the path
marked in red. The rover’s on-board localisation system using dead reckoning or a VO
type system attempts to estimate the rover position, and produces an estimated path
marked in blue. After a certain length of traverse, the rover stops to take a series of
images, to construct a 360 degree panorama. A feature detection system attempts to
detect large boulders in the scene, filtering out those boulders that would be too small
to be observed in orbital imagery. The centroids of these features can be mapped to
used to produce a top down local feature map, mapping the boulders highlighted in
green in Figure 2.15a to the first map shown in Figure 2.15c. Using the rover’s estimate
of its pose (the end point of the blue traverse), together with the confidence associated
with that estimate an elliptical search space can be defined on the planetary surface as
shown in Figure 2.15b. This ellipse acts as a search space in the pre loaded map, which
has features derived from orbital imagery. This data will have been computed and
loaded onto the rover before launch. The area within the ellipse in the orbital map can
be searched for patterns of large boulders which match the arrangement of the boulders
in the local map. The large features in the orbital map search space are highlighted in
blue in Figure 2.15b and shown as a point map in the bottom map in Figure 2.15c. By
comparing the two maps in Figure 2.15c using the ICP method discussed earlier, the
rotation and translation between the two maps can be calculated and used to relocalise
the rover in the global map. This would provide absolute localisation for the rover in
a global map.

Two architectures have been developed to implement absolute localisation of a plan-etary rover by using a global map derived from orbital imagery and a local map generated
from images taken by the rover during its traverse on the surface. Both methods create
an offline map of features, which is pre-loaded onto the rover memory. The rover uses
online feature detection during its traverse to make a local map based on its relative
localisation, which is then compared with the orbital map to aid in localising the rover.
These methods are opportunistic, in that they both depend on the rover encountering a
sufficient number of large boulders (large enough to be detected by an orbital platform)
over the course of a long traverse.
Figure 2.15: Planetary Constellation Matching localisation system [91]. (a) The rover traverses from the starting point at the bottom of the map. The green line represents the planned path. The red line represents the actual path traversed by the rover, including deviations due to obstacle detection and also wheel slip. The blue line represents a localisation estimate from a VO or SLAM system. After a fixed traverse (e.g. 15 metres), the rover takes a panoramic image of its surroundings and identifies large features (>1m diameter), which have been highlighted in green. (b) shows the search area in the orbital map, centred on the rover’s estimated location which is also searched for large features, which have been highlighted in blue. (c) The top graph shows a top down view of the rover’s local feature map, derived from the panoramic imagery with the rover position (the map origin) represented by the black dot. The bottom graph shows a top down view of the global map, derived from the orbital imagery. Corresponding points in these two maps are circled in red, and would be co-registered using an ICP algorithm.
2.4. Orbiter Image Localisation

<table>
<thead>
<tr>
<th>Platform</th>
<th>Class</th>
<th>Geometric constraint</th>
<th>Texture constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rover</td>
<td>Rock</td>
<td>&gt; 0.6m² and height &gt; 0.25m</td>
<td>H: &gt; 2 · σ with E: &lt; −2 · σ</td>
</tr>
<tr>
<td></td>
<td>Outcrop</td>
<td>&gt; 5.5m²</td>
<td>H: &lt; −2 · σ OR E: &lt; −2 · σ</td>
</tr>
<tr>
<td>Orbiter</td>
<td>Rock</td>
<td>&gt; 0.6m²</td>
<td>H: &gt; 2 · σ</td>
</tr>
<tr>
<td></td>
<td>Outcrop</td>
<td>&gt; 8m²</td>
<td>H: &lt; −2 · σ</td>
</tr>
</tbody>
</table>

Table 2.9: Geometric and texture constraints for feature classification in [16]. H represents the distribution of the eigenvalues of the Hessian. E represents the distribution of the local entropy.

The method described in [16] uses the same method for identifying, extracting and classifying features in both the global (orbital) and local (rover) maps. The Hessian response of the image is used to identify blob-like ROIs in the image. This is combined with a weighted measure of the local entropy (or the "randomness") of the area around each pixel \( p(i) \), calculated using Equation (2.29), where \( \mu \) is the mean of the pixel’s neighbourhood.

\[
LE^w = -\sum_{i=0}^{255} w(i) \cdot p(i) \cdot \log_2(p(i))
\]

\[
w(i) = 1 - G(\mu, \sigma_e)
\]

\[
\sigma_e = 8
\]

To build the local map, the rover needs to perform relative localisation. This is performed by fusing data from an IMU with VO localisation estimates, based on 3D stereo reconstructions. The detected features are classified as either Rocks or outcrops in order to facilitate matching between the local and global maps. The classification algorithm uses both geometric and texture based metrics. The geometric measures are fixed for each class (e.g. a Rock in the orbital map must have an area greater than 0.6m²), whereas the texture based measures are adaptive based on the distribution of the local neighbourhood of each pixel. The features are classified as either individual rocks, or larger rocky outcrops.

The feature matching process between local and global maps is based on the method
in [19], however instead of LIDAR the local features are generated from stereo-vision based 3D reconstruction. Furthermore the comparison is performed using 2D maps rather than 3D maps in order to simplify the calculation and the computation. The flattening of the maps can have the effect of reducing the accuracy of the local map (as feature points have to transposed to a ground plane), however as the orbital maps are 2D maps the comparison becomes simpler. The matching of feature points uses the DARCES algorithm [20], which takes the feature centroids and creates triangles using them. The two sets of features from the global and local maps, are matched by finding the arrangement of matching triangles that minimises the error between them.

The second architecture [62] uses separate methods for generating the global and local map. The global map is generated by first applying Otsu’s thresholding method [77] to the orbital images. This produces an initial, noisy binarised feature map. A median filter is applied across the image, sized so as to remove noise, without compromising the edges of features. Remaining blobs with areas below a certain threshold are rejected at this stage, and the remaining blobs form the global feature map.

The rover produces an estimate of relative localisation using VO, again combined with IMU data. Images captured by the rover’s onboard stereo cameras are used to extract boulders to create the local boulder map. The images are initially segmented using mean shift [23], a technique based on iterative clustering, which works robustly with clusters of arbitrary size, shape and number. Clusters are merged in order to reduce over segmentation, and very small regions are rejected. Outlier rejection is performed in 3D space, by comparing the tracked position of features between successive frames.

The matching process between global and local maps is performed using the ICP algorithm. This allows the in-plane rotation and transform between the local map (based on the relative localisation estimate) and the global, absolute map from the orbiter. The full set of points in the orbital map is not used, rather a local area around the rover is used in order to reduce the computational complexity. Poor matches are filtered out to improve the performance of the matching.
3.1 Introduction

The motivation behind the development of Planetary Monocular Simultaneous Localisation And Mapping (PM-SLAM) was to create a computationally lightweight, robust, monocular vision based SLAM for planetary exploration, that could be utilised in missions with long linear traverses. To this end each of the component parts of the system...
have been designed to function using limited resources in a challenging environment. Due to the nature of planetary missions, the robustness of the system was also a critical driver.

SLAM is a probabilistic technique for estimating the position of a mobile agent in an unknown environment while concurrently creating a map of local features. As explained in Section 2.2.1, this is achieved using a combination of wheel odometry used with a model of the vehicle dynamics and environmental observations from onboard sensors. Both the odometry and the environmental observations are assumed to be inherently noisy; a margin of error exists in the accuracy of the odometry (e.g., due to wheel-slip) and accuracy of the environmental observations depends upon the sensors used. Combining these sources of information can minimise their associated errors.

SLAM is commonly implemented using Gaussian estimators (e.g., Extended Kalman Filter described in Section 2.2.1 and the Extended Information Filter described in Section 2.2.2) which was found to provide better accuracy in sparse, largely homogeneous environments. An alternative to these are particle filters, such as FastSLAM.

### 3.2 Structure of PM-SLAM

A brief overview of PM-SLAM follows. The system is composed of a combination of image processing and feature extraction modules together with a SLAM filter module. A block diagram of the system is shown in Figure 3.1.

The two main inputs to the system are the odometry or control signal and monocular images. The former is normally provided as a dead-reckoning estimate of the rover motion, either from wheel odometry or from the control signal used to command the rover to move. The movement reported by this input is assumed to be noisy, and is used by the predict step of the SLAM system. For each control signal passed to the system, an image is expected. This image represents the rover’s view of the local scene after executing the movement represented by the control signal. Features extracted from the image inputs are used in the update and predict steps of the SLAM system.
3.2. Structure of PM-SLAM

Usually control data, whether in the form of odometry or control commands, are generated at a much higher rate than images from an onboard camera, as cameras typically take longer for to generate data. It is also not desirable to generate large amounts of image data on board a planetary rover platform, as computer memory for storing it is limited. Therefore control data is accumulated in between image captures.

The control and measurement data are assumed to be calibrated. During construction the sensors are carefully calibrated and upon landing, it is common for the scientific and engineering cameras on a planetary rover to be calibrated using plates with known features attached to the rover body as well as other known values [13]. The SLAM element of the GNC does not directly drive the locomotive system, nor does it directly control the images streamed from the camera, therefore it is assumed that any bias in the signals have been filtered before being presented to PM-SLAM.

In order to make sure that the accumulated odometry reading and the corresponding image from the camera are correctly paired when sent to the SLAM filter, a synchronisation module is used. For each rover step (e.g. a regular fixed distance of traverse), the SLAM filter module is first passed the odometry data which is used to propagate the estimate of the rover pose (the predict step). At the same time the image feature extraction and matching module handles the extracting of features from the images.

The image processing and feature extracting module takes a raw monocular image as input, and outputs a list of features located in the image, giving them an identifier, an estimate of the 3-dimensional real-world position of the feature in the rover frame and a flag indicating if the feature has been detected before. This is achieved using two novel methods introduced in this paper, hybrid-salient feature tracking and direct depth.

The hybrid-salient feature tracking system (described in section 3.3) extracts feature bounding boxes from the image and determines if they have been previously detected, keeping a database of all detected features.

The real-world rover-frame position of the features is calculated in parallel, using the a depth perception module implementing the direct depth algorithm (described in section 3.4). The data from the feature matching module and the depth module are merged
and the features are passed to the SLAM filter. The SLAM filter then updates the rover position and map estimates (the update and augment steps).

PM-SLAM’s modular design is implemented in C++ using the ROS framework. ROS is a commonly used platform in the robotics community, and provides a cross platform, threaded environment for developing complex robotic applications [82]. The architecture allows modular units (nodes) to pass each other sensor data (and other robotic system data).

PM-SLAM utilises ROS services, which facilitate request/response communication between independent nodes. Using service allows separation of the various functional elements of the system while ensuring they can communicate easily. Services were chosen (over the alternative publish/subscribe topic model) to ensure the synchronisation of data across several parallel nodes, which each require different amounts of time to execute.

3.3 Feature Detection and Tracking

3.3.1 Introduction

In order for a vision-based SLAM system to be able to reliably estimate a rover’s position on the planetary surface, detection and tracking of the visual features used as input to the SLAM filter must be performed robustly. The system should avoid mismatching features, i.e. it should not identify a visual feature incorrectly as a feature observed previously in the rover’s traverse. In the case of a mismatch occurring, SLAM will attempt to compare the positions of two different landmarks in order to determine the rover’s relative localisation. As two separate landmarks are being compared, this comparison will provide false information to the filter, and will reduce the accuracy of the localisation estimate. The tracking of visual features in a planetary environment is especially challenging as the visual scenes encountered tend to be homogeneous and unstructured. In addition the locations a rover will traverse will not have been previously explored, and hence there is a level of uncertainty as to what will be observed. The unknown nature of the environment also makes development of top-down visual systems unsuitable, as training datasets cannot be compiled prior to a mission.
3.3.2 Visual SLAM Using Point-Based Features

Terrestrial visual SLAM systems commonly use point-based visual features, such as those described in Section 2.3.1. SURF is a robust scale and orientation invariant system that has been used in several SLAM implementations \[5, 30, 44\]. The dynamic distance metric between descriptors allows SURF to perform well when matching features in varied conditions. The use of SURF as an input to a SLAM system tends to yield a large number of features per image (in the case of SLAM at every detection step). Having lots of features can be beneficial, providing plenty of trackable features to the SLAM system, which helps to improve the SLAM estimate. However large amounts of features also burden the SLAM filter, in terms of the computation needed to process them all. For instance in the Gaussian filters, such as the EKF, each feature’s global position in the map, along with the associated covariances are stored in the state vector and the covariance matrix of the filter. As these grow, steps that involve complex matrix operations (such as inversion), require increased time to be performed.

The number of features output by SURF can be adjusted by varying various parameters of the algorithm, for instance the initial Hessian size, used to perform corner detection, can be adjusted. A larger starting Hessian increases the corner detector size and produces a smaller number of features, whereas a smaller starting Hessian value will produce a large number of features. This can be seen from Figure 3.2.

Reducing the number of features detected by SURF drastically decreases the processing time required for each SLAM step, as it limits the rate at which the SLAM state grows. However, if fewer features are detected per step, each individual feature point will have an increased weight when passed to the SLAM filter, including potential mismatches. The lower number of features would also decrease the chances of the same features being re-observed over several frames, reducing the amount of useful information being passed to the SLAM filter and also compromising its ability to loop-close. These factors combine to decrease the accuracy of the SLAM position estimate.

Figure 3.3 shows the results of running SLAM using SURF features as input to an EKF filter. The system was run with the initial Hessian size set to 400, 12000 and 20000 for each run. An image dataset was generated for 660 steps using the PANGU
Chapter 3. Planetary Monocular Simultaneous Localisation And Mapping

Figure 3.2: Number of features generated using SURF algorithm with varying initial Hessian size. SURF keypoints are represented by red dots.
3.3. Feature Detection and Tracking

Figure 3.3: Performance of SLAM using SURF features. The blue line shows the rover pose error increasing with the initial Hessian size, while the red line shows that the average duration of each SLAM step decreases.

simulator. The traverse distance was a D-shaped loop covering a total of 479 steps, covering 60.1 metres. For each step the rover pose was recorded and the L2 norm from the groundtruth position was calculated. The average of this distance measure was recorded as the average rover position error. The duration of each SLAM step was also recorded, which included only the time required to perform the SLAM filter operations (predict, update and augment), and excludes the time required to perform the SURF detection, extraction and matching, which is negligible in comparison, and does not change significantly with initial Hessian size.

The graph shows a clear trade-off between accuracy and required computation time for SURF based SLAM, i.e. by adjusting the Hessian it is possible to have a fast, inaccurate SLAM or a slow accurate SLAM.

3.3.3 Visual SLAM Using Saliency-Based Features

An alternative form of feature that can be used as an input for vision-based SLAM systems are salient features (as described in Section 2.3.2). The output of salient feature detection systems is in the form of a set of regions of interest in the image, or “blobs”.

The larger size of the blobs means that generally fewer features are detected in a similarly sized image than if point-based features are used.

In the case that one landmark occludes another in the visual scene, or two landmarks are very close together, visual saliency techniques can tend to have trouble differentiating them, and will output a single blob for several features. This can cause issues for depth perception techniques that place the detected visual features, as the bounding box used will represent more than one real world feature. Problems can also occur if after movement, the rover’s perspective has changed enough to enable the previously grouped landmarks to be individually detected.

Saliency features can be tracked using k-NN algorithms, selecting the nearest blob in consecutive frames. This method of tracking means that features that have not been observed in consecutive frames cannot be re-identified. Using this method of tracking does not allow SLAM systems to loop-close.

### 3.3.4 Hybrid-Salient Feature Detection and Tracking

The salient feature tracking technique takes the features generated by the saliency detection techniques discussed previously, and associates them with point-based descriptors detected within their bounding boxes. This combination of a salient feature bounding box with a set of point-based feature descriptors is referred to as a *hybrid-salient feature*. These hybrid-salient features can then be tracked across successive images by matching their associated point-based features. This ensures that a large number of features can be used by the tracking and matching algorithm, but these points are effectively clustered to reduce the amount of points passed to the SLAM filter, i.e. one per landmark. The robustness of the tracking potentially increases with each successful tracking, if new point-based descriptors are associated with the matched feature. The associated point-based descriptors also allow salient features to be tracked across non-consecutive images. The hybrid-salient feature tracking system has been designed in a modular fashion to allow the combination of any saliency algorithm with any point-based feature extractor. This modular structure is apparent in the design of the data-pipeline shown in Figure 3.4. The first step is to use a salient feature detection
3.3. Feature Detection and Tracking

Figure 3.4: Hybrid-salient feature detection and tracking data pipeline.

algorithm, such as those discussed in Section 2.3.2, to identify regions of interest in the image. Figure 3.5b shows binary blobs extracted from the example image shown in Figure 3.5a. The blobs are used to create bounding boxes, which are used to define regions of interest in the subsequent steps, however bounding boxes which have an area lower than a threshold are discarded. This threshold is selected as a percentage of the entire image. A point-based feature keypoint detector, such as those in Section 2.3.1, is used to extract feature points within each bounding box as shown by the white circles in Figure 3.5d. Bounding boxes which contain fewer keypoints than a threshold are discarded before the matching step, as a low number of features will not give a confident match. In the final step the point-based features are used to match the bounding boxes against a database of hybrid-salient features from previous iterations of the algorithm. Figure 3.5e shows the result of this matching, with features successfully matched with a previous feature marked with a blue bounding box and those without a match marked with a red bounding box as new features. Any new keypoints associated with previously detected hybrid-feature are added to the corresponding hybrid-feature in the database. New hybrid-features are added to the database along with their keypoints for future matching.

In order to perform the tracking of hybrid-salient features a confidence metric ($M$) is used to compare a hybrid-salient feature from the current image with one that has been
Figure 3.5: An example showing each step of hybrid-salient feature tracking: (a) Rectified image from SEEKER dataset, (b) Saliency map, (c) Bounding boxes extracted from saliency map. Bounding boxes that are too small or do not contain enough point-based features have been discarded (the small bounding box on the right of the image). (d) Point-based features extracted from within the bounding boxes, (e) Hybrid-salient features matched against database, with new features in blue, previously observed features in red with their associated feature ID.
previously observed. A higher confidence metric implies a higher confidence that the
two features represent a re-observation of the same real world object. The confidence
metric is constructed by computing the Euclidean distance metrics of the associated
point-based descriptors:

\[
M(i, j) = \{ v \in [0, 1] \subset \mathbb{R} | i \in D \subset \mathbb{N}_{>0}, j \in C \subset \mathbb{N}_{>0}, v = G_{ij}(Q_{ij}) \} \tag{3.1}
\]

where \( D \) is the set of all databased salient feature indices, \( C \) is the set of all current
frame salient feature indices and where:

\[
G_{ij}(q_{ij}^1, \ldots, q_{ij}^n) = \{ v \in [0, 1] \subset \mathbb{R} | 1 \equiv P_{ij} \subset \mathbb{N}_{>0}, q_{ij}^\alpha \in Q_{ij} \} \tag{3.2}
\]

where \( Q_{ij} \) is the set of Euclidean distances of point-based descriptors from nearest
neighbour matches between \( i \) and \( j \) and \( P \) is the set of all indices of point-based feature
matches between \( i \) and \( j \). \( M \) and \( G \) are designed such that:

- \( G(q_{ij}^\alpha) \rightarrow 1 \) when \( q_{ij}^\alpha \rightarrow 0 \);
  so that a high confidence pixel-based feature match would cause a high confidence metric;

- \( G(q_{ij}^\alpha, q_{ij}^\beta) \rightarrow G(q_{ij}^\alpha) \) when \( q_{ij}^\beta \rightarrow \infty \);
  so that outliers are removed;

- \( G(q_{ij}^\alpha) \rightarrow 0 \) when \( q_{ij}^\alpha \rightarrow \infty \);
  so that a low confidence pixel-based feature match results in a low confidence metric;

- \( G(q_{ij}^\alpha, q_{ij}^\alpha) > G(q_{ij}^\alpha) \) when \( q_{ij}^\alpha \rightarrow 0 \);
  so that a more high confidence pixel-based feature match results in a higher confidence metric;

- \( G(q_{ij}^\alpha, q_{ij}^\alpha) < G(q_{ij}^\alpha) \) when \( q_{ij}^\alpha \rightarrow \infty \);
  so that a more low confidence pixel-based feature match results in a lower confidence metric;
To achieve these criteria, the function $G$ is defined as:

$$G_{ij}(q_{ij}^1, ..., q_{ij}^\alpha, ..., q_{ij}^n) = 1 - \frac{1}{1 + \frac{1}{t_{ij}^1} + ... + \frac{1}{t_{ij}^\alpha} + ... + \frac{1}{t_{ij}^n}}$$  \hspace{1cm} (3.3)$$

where $t_{ij}^\alpha = a \cdot q_{ij}^\alpha + c$; and $a \in \mathbb{R}_{>0}$ and $c \in \mathbb{R}$ are constants. This function is structured so that as the Euclidean distance between feature descriptors tends to 0 (i.e. the two point based features are considered very similar), the confidence metric tends to 1, where as when the Euclidean distance between the descriptors tends to infinity (i.e. the two point based features are considered very dissimilar) the confidence metric tends to 0. Therefore the result of the calculation gives a confidence close to 1 for confident matches and closer to 0 for non-confident matches. The values of $a$ and $c$ are selected to normalise the values of the Euclidean distances between the descriptors for each of the different point-based feature matching algorithms. The range of Euclidean distance values produced by SURF (descriptor described in Section 2.3.1.2), SIFT (descriptor described in Section 2.3.1.1) and BRISK (descriptor described in Section 2.3.1.3) are not equivalent, and must be scaled in order to produce a consistent confidence metric when using different configurations of PM-SLAM. The calculation of the Euclidian distance is calculated as:

$$q(a, b) = \sqrt{\sum_{l=1}^{L} (a_l - b_l)^2}$$  \hspace{1cm} (3.4)$$

where $a$ and $b$ each represent a point based feature descriptor vector produced by SURF, SIFT or BRISK. $L$ represents the size of the descriptor vectors (64 for SURF, 128 for SIFT and 512 for BRISK).

The implementation of this is described in Algorithm 1.

As Figure 3.6 demonstrates, point-based feature detection and matching is prone to generating outliers. Hybrid-feature detection enables fast and easy outlier rejection and robust matching and tracking.
Figure 3.6: Demonstration of outlier rejection difference between traditional point-based feature matching and hybrid-salient matching. (a) An example of traditional feature matching between two sequential SEEKER images using SURF and FLANN. Two consecutive images are shown side-by-side. The features are represented by white circles and correspondences between the two images are shown by lines connecting co-registered points between the two images. (b) An example of feature matching between two sequential SEEKER images using Hou and SURF hybrid-saliency implemented in PM-SLAM. Two consecutive images are shown side-by-side. The features are labelled with their associated feature ID, with the bounding boxes of new features shown in blue and previously observed features shown in red. IDs with the same number are matching images across the consecutive images.
Algorithm 1: Hybrid salient feature matching algorithm

1. function MatchFeatures(currentFrameBoundingBoxes,
                             databaseOfHybridSalientFeatures)

   2. foreach boundingBox in currentFrameBoundingBoxes do
      3. Extract point based features from currentFrame using boundingBox
      4. Find matches between point based features and
         databaseOfHybridSalientFeatures
      5. foreach match in listOfMatches do
         6. Place matches in bins according the index of the database Hybrid Salient
            feature matched
      7. foreach bin in bins of matchedHybridSalientFeatures do
         8. Calculate confidence metric for each matched hybrid salient feature
         9. The index of the hybrid salient feature with the highest confidence is set as
            the match The point descriptors from the current frame are added to the
            matched hybrid salient feature in the database

3.3.5 Evaluation of Hybrid-Salient Features

In order to validate the hybrid-feature detection and tracking in PM-SLAM, the system
was tested independently using a subset of the SEEKER dataset, separate from the
dataset used in the analysis of PM-SLAM as a full system. The metric used to evaluate
the system performance is Multiple Object Tracking Accuracy (MOTA). MOTA is a
measure which is commonly used in vision processing applications where successful
tracking of objects is required [51]. MOTA is defined as:

\[
\text{MOTA} = 1 - \frac{\sum (m + fp + mme)}{\sum g},
\]

(3.5)

where \( m, fp, mme \) are the number of misses, false positives, and mismatches respectively; and \( g \) is the total number of detections.

In order to determine the MOTA of the hybrid-saliency feature detection system a
dataset was compiled using an annotation tool developed in house. Human operators
3.3. Feature Detection and Tracking

Table 3.1: MOTA results for detection and tracking using hybrid-saliency on SEEKER data

<table>
<thead>
<tr>
<th>Number of frames</th>
<th>MOTA</th>
<th>Mismatches</th>
</tr>
</thead>
<tbody>
<tr>
<td>141</td>
<td>0.97238</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.1: MOTA results for detection and tracking using hybrid-saliency on SEEKER data were shown the SEEKER images in sequence, and in a two step process identify the elements in the visual scene. In the first step, shown in Figure 3.7a, the user is required to draw bounding boxes around areas they deem to contain features (i.e. rocks and boulders). Once all of the features in the scene have been marked the annotator is shown the previous image along with the tracked IDs of the features, as well as the current image with the bounding boxes numbered in the order they were entered (as shown in Figure 3.7b). The operator can then assign an ID to each bounding box in the scene, by either giving it the same ID as the object had in the previous frame or by assigning it an ID one greater than the total number of features recorded up to that point.

A set of 141 sequential images from a traverse were used to evaluate both the detection and tracking elements of the hybrid-saliency feature matching. The results of this evaluation are shown in Table 3.1.

A larger subset of 1136 frames was used to evaluate only the tracking element of hybrid-saliency (i.e. the ability to correctly identify a blob across successive, but not necessarily consecutive frames). The blobs were provided to the system by the annotation bounding boxes generated by the human groundtruthing. As well as the MOTA parameters, the step time was also recorded for each of the hybrid-saliency detectors combinations used. The results are shown in Table 3.2 and Figure 3.8. The overall MOTA was used to show the robustness of the tracking of the system, and the number of mismatches was also recorded as this is an important parameter in the context of SLAM systems.

The results shown in Table 3.2 indicate that hybrid-feature detection and tracking performs with high level of accuracy. All of the hybrid-saliency feature detection combinations produced MOTAs above 91.5%, which is a very robust performance. Moreover, the small number of mismatches is of great significance to the effectiveness of technique, as passing mismatches to the SLAM filter can significantly affect the estimated...
Figure 3.7: Steps of annotation tool used to generate feature tracking data for SEEKER dataset. (a) Selecting features in scene. (b) Assigning feature IDs to recorded features.

<table>
<thead>
<tr>
<th>Detector</th>
<th>MOTA</th>
<th>Mismatches</th>
<th>Total Detections</th>
<th>Avg. Step Time (s)</th>
<th>Total Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hou SURF</td>
<td>0.923</td>
<td>38</td>
<td>4086</td>
<td>0.232</td>
<td>264.641</td>
</tr>
<tr>
<td>Hou SIFT</td>
<td>0.985</td>
<td>77</td>
<td>4071</td>
<td>0.854</td>
<td>970.504</td>
</tr>
<tr>
<td>Hou BRISK</td>
<td>0.915</td>
<td>41</td>
<td>4108</td>
<td>2.272</td>
<td>2581.331</td>
</tr>
</tbody>
</table>

Table 3.2: MOTA results for tracking using hybrid-saliency on SEEKER data
3.3. Feature Detection and Tracking

![Results of hybrid-saliency tracking evaluation using 1136 SEEKER images.](a)

- **Figure 3.8:** Results of hybrid-saliency tracking evaluation using 1136 SEEKER images. 
  
  **(a)** Average MOTA for traverse as far as the current traverse step, shown on the x-axis. 
  
  **(b)** Time taken to perform detection and matching for each step.
As Figure 3.8a shows, the MOTA values for all three combinations very quickly became stable, producing a consistent range of values after the initial 200 or so steps. This means that as the internal database of tracked blobs and their associated point-features and descriptions grew with each step, the performance of the system was not affected, and continued to produce robust tracking.

The time taken for each detection and matching step showed a slight increasing trend throughout the complete traverse, visible from the curves in Figure 3.8b. This is a consequence of having more feature points to check against each other at the descriptor matching phase of the algorithm. Hybrid-saliency using SURF descriptors was the combination whose computation time stayed consistent throughout the entire traverse. This is likely due to the fact that SURF descriptors are smaller than SIFT descriptors, and SURF point comparisons make use of a Laplacian directionality flag as a first step in matching. This drastically reduces the subset of features that have to be compared using their full descriptor. For all of the combinations the number of features in each scene was a more significant indicator of the length of the step time, than the number of steps already completed (which in turn is related to the size of the feature database).

The BRISK detection was much slower than the other two algorithms investigated. This is due to the number of point-based features generated by the BRISK algorithm. The larger number of features produced means that during the feature matching step more comparisons need to be carried out to find matching hybrid features, which in turn slows down the matching process.

Mars rover missions to date have all had limited operating hours, with traverses limited to approximately 2 hours around midday. This restriction is due to power constraints for solar powered platforms, as well as the limitations of sensors (such as cameras) operating in sub-optimal lighting conditions. The rovers also tend to traverse a limited region of latitudes, meaning the lighting environment is unlikely to change significantly over the traverse. This means it is unlikely that any visual feature detection system will be required to detect and track features across vastly different lighting conditions, however for the sake of robustness a brief experiment was carried out.
3.4 Monocular Vision-Based Depth Perception

PM-SLAM has been designed to utilise monocular images, with an aim to creating a computationally lightweight but still robust localisation system that could be used on
lower specification hardware, such as those used in planetary exploration and micro-rover design. Removing the computationally complex step of generating depth images from stereo pairs could improve the traverse speed of a rover platform \[91\]. However, the use of monocular images makes depth perception (an important element of SLAM) challenging \[33\]. Terrestrial methods such as Inverse Depth Parameterisation \[22\], increase the number of parameters representing the each feature’s state, and as the goal is to limit the complexity of the SLAM computation a larger state would be antithetical to the design motivations for PM-SLAM. PM-SLAM uses a geometric method (known as, Direct Depth) in order to achieve depth perception from monocular images within the current framework. This removes the need for expensive operations such as correlation of points between stereo image pairs. The reduction in computational complexity would be a factor in helping to achieve the traverse speed and accuracy specified in Section \[1.2\].
3.4. Monocular Vision-Based Depth Perception

Direct depth is an extension of the method employed in [18]. In order to deal with the scale ambiguity of detecting the distance of an object of an unknown scale in a monocular image, the following assumptions are made:

- the width $U$ and height $V$ of the camera images are known, in pixels.

- the camera’s Vertical Field Of View $VFOV$ and Horizontal Field Of View $HFOV$ angles are known.

- the camera viewable region is a flat plane relative to camera mounting.

- the height and pitch of the camera $h$ and $\alpha$ are known relative to the flat plane (as in Figure 3.11).

Due to the risk of catastrophic failure, planetary rovers are designed to only operate within regions of relative safety. Landing sites and exploration targets are selected for the ease of navigability. Furthermore, any navigation cameras would be given a high pitch angle in order to survey the area directly in front of the rover. For instance a 65 degree field of view camera, with a 30 degree pitch angle mounted 2 metres above the ground, the furthest visible terrain would be 5 metres away. It would be unlikely that the rover would be presented with scenes with large slopes in the 5 metre maximum field of view. Therefore it can be assumed that for local camera views the first condition will
These parameters would need to be calibrated and confirmed after landing the rover on the planetary surface, as part of an initial camera calibration mission phase. Existing missions such as the MER missions use extensive calibration to ensure the accuracy of the captured images, as well as any parameters of the camera user of photometrics \cite{13}. If a calibration step was not used to verify the height $h$ and pitch of the camera $\alpha$, then the resulting depth estimates from the technique described in this section would not be sufficiently accurate to use for feature detection and tracking. The specific characteristics of the motors and physical mechanisms used to position the camera and mast would inform the uncertainty model for a specific platform, resulting in a discrepancy between reported and actual height $h - h_e$ and pitch $\alpha - \alpha_e$. The parametric uncertainty would be propagated through the equations in \ref{equation:3.9} and \ref{equation:3.7}.

The $y$ position in the world frame is calculated from the pixel position $(u,v)$ in the image frame, (made considerably easier by the fact that $x$ axes in both the world and image frames are parallel unlike the $y$ axes). Where the axes are defined as in Figure \ref{figure:3.11} The angle $\beta$, which is defined as angle around the $x$ axis with the origin at the
3.4. Monocular Vision-Based Depth Perception

camera, is calculated as:

\[ \tan(\beta) = \frac{2v}{V} - 1 \tan\left(\frac{VFOV}{2}\right) \]  \hspace{1cm} (3.6)

Using simple geometry:

\[ y = \frac{h}{\tan(\alpha + \beta)} \]  \hspace{1cm} (3.7)

To calculate \( x \) in the world frame, the angle \( \gamma' \) (defined as the angle at the camera origin and around the normal to the plane \( P \), and parallel to both the \( x \) axis and the line \( L \) between the camera origin and the feature in camera frame) (Figure 3.12) is used.

\[ \tan(\gamma') = \frac{u - U}{\sqrt{\left(\frac{U}{2\tan(HFOV/2)}\right)^2 + v^2}} \]  \hspace{1cm} (3.8)

Then using right-angled triangle formed on the plane \( P \) with the shared camera origin, the feature on the world frame and point on both \( P \) and the world frame which is closest to the camera origin, \( x \) can be calculated:

\[ x = (\sqrt{h^2 + y^2}) \tan(\gamma') \]  \hspace{1cm} (3.9)

The direct depth equations are non-linear, and therefore are prone to non-linear error. Figure 3.13a gives an example of the distance calculations for each pixel along the \( y \)-axis of the image (from top to bottom), with a square FOV of 65 degrees, a tilt of 30 degrees and a 512 x 512 pixel image size taken from a camera mounted at 2m. The real world change in distance for each single pixel change in position is much larger towards the top of the image, than it is towards the bottom of the image. This means a single pixel error in determining the location of a feature in the image plane towards the top of the image creates a very large difference in real-world \( y \) position of the feature. While differences in the \( x \)-axis of the image produce less pronounced errors in the real-world co-ordinate frame, the real-world \( x \)-axis location is dependent on the calculated real-world \( y \)-position, and so is also sensitive to errors in the \( y \)-position of features in the
Figure 3.13: Graphs showing relation between image-plane y pixel position and real-world (a) y-axis position and (b) x-axis position.
3.4. Monocular Vision-Based Depth Perception

Figure 3.14: Illustration of the differences in error for direct depth y-axis estimates using the centroid and bottom edge of the feature bounding box. The rover assumes a flat ground plane, shown as a dotted line, but the actual terrain has slight variation in height. The boulder appears in the image plane bounded by the green and red lines. The blue line represents a ray from the centre of the bounding box to the rover’s ground plane. The blue and green dots represent the real world positions of the centroid and bottom edge of the image plane bounding box respectively. The rays through these points are continued down to the ground plane, representing the direct depth estimate. The difference between these points on the y-axis represent the depth errors, $e_c$ and $e_b$ for the centroid and bottom edge.

The direct depth estimate for points which lie significantly above the ground plane will have a larger error associated with them compared with objects closer to the floor, whose error should only represent any change in elevation of the actual terrain. The centroid of the bounding boxes produced by hybrid-saliency represents a point halfway up the visible face of the feature in the rover. This point is likely to be elevated above the ground. In contrast the bottom edge of the bounding box will in the majority of cases represent the bottom of the front face, where the feature makes contact with the terrain and therefore would have a smaller error when projected down to the rover ground plane by the direct depth algorithm. These two cases are illustrated in Figure 3.14.
A sequence of ten images generated using the Planetary and Asteroid Natural Scene Generation Utility (PANGU) were generated, using a template developed by Airbus Defense and Space for testing the Exomars rover. These images were used to give a real world demonstration of this error effect. A single boulder was selected and a bounding box was drawn around the feature in each image frame, and in each frame the “rover” moved 1 metre closer to the boulder. The centroid and bottom centre of each of the annotated bounding boxes from the images were passed into direct depth at each step and a depth estimate was computed for each of them. As PANGU renders images using a full 3D model of the terrain, the exact position of each image plane pixel can be determined in real world co-ordinates (i.e. its position in the model). The actual positions of the bottom and centroid of each bounding box was retrieved from PANGU and the error between the direct depth estimate and the actual position were calculated. The results of this experiment are shown in Figure 3.15.

The error was much smaller when the bottom of the bounding box was used. Although
the error grew large for both the centroid and bottom centre of the bounding box as the
distance to object increased, these distances were outside of the range where direct-
depth would be used in PM-SLAM. The PM-SLAM implementation of direct depth
uses the bottom (y-axis) centre (x-axis) image co-ordinates of the bounding boxes of
the extracted features as a result of this analysis.

The simplicity of the geometric model used in direct depth allows for computationally
efficient calculation of feature depth using only monocular images. While the system
makes several assumptions about the local environment, these can be justified within
the parameters of a planetary exploration mission.

3.5 Implementation

PM-SLAM is implemented in C++ using a modular design. Each component of the
system has been implemented as ROS node to facilitate simple and scalable commu-
nications between various parts of the system as well as allowing for a ”plug and play”
structure.

The co-ordination module co-ordinates the all the input data, i.e. the control signal
and the camera images. It syncs these two data streams and provides them to the
rest of the system. This module also provides the configuration for which combination
of saliency, point-based and SLAM filter algorithms are used by the system. These
settings were changed using a configuration file loaded at startup.

Three SLAM modules were implemented, the EKF, the EIF and FastSLAM2. The
EKF was implemented based on the MATLAB filter by [6]. The vehicle model, control
parameters, measurement model and feature parameters have been altered to reflect
the models described in Sections 2.2.2 and 2.2.3. The computational efficiency of the
EKF has been improved by using Cholesky Decomposition to find the square root of
matrices [103].

The EIF was implemented based on the MATLAB ESEIF filter by [6]. As with the
EKF the models were updated so as to reflect the vehicle and measurement models
described in Sections 2.2.2 and 2.2.3.
FastSLAM 2 has been implemented based on the MATLAB FastSLAM filter by [6]. As with the EKF and EIF the vehicle and measurement models were updated to those in Sections 2.2.2 and 2.2.3.

The initial parameters provided to the filter algorithms by the launch files are the starting covariance of the filter, the control noise in the x direction, y direction and heading, and the measurement noise for the range and bearing. These noise values depend on the specific sensors and platforms used to collect the data. For PANGU generated datasets these values were set when creating the dataset. For other datasets, where the values were not available a small subset of the data was used to empirically tune them. The tuning subset was not used in the final experiments.

The hybrid saliency node of PM-SLAM is comprised of several salient feature detection algorithms, several feature point-based feature extraction modules and a hybrid feature matching module.

The Hou salient feature extraction algorithm has been implemented in C++ based on the description of the algorithm in [45]. The module takes an input image and outputs a set of bounding boxes indicating regions of interest.

The Rudinac algorithm was also implemented in C++ based on the description of the algorithm in [86]. As with the Hou algorithm implementation it takes an image as input and outputs region of interest bounding boxes.

SURF point-based feature extraction uses the OpenCV implementation with the region of interest taken as the salient feature and the starting octet size.

The SIFT point-based feature extraction uses likewise uses the OpenCV implementation of SIFT. In order to perform feature comparison, the matching of SURF and SIFT algorithms uses the OpenCV implementation of Fast Approximate Nearest Neighbour Search Library (FLANN) for finding closest descriptor matches, which is based on the algorithm in [76].

BRISK point-based feature extraction was also implemented using an OpenCV implementation. Descriptor comparison for feature matching is performed using a brute force L2 algorithm which is part of the OpenCV libraries. BRISK has a binary descriptor so cannot utilise the FLANN algorithm.
The hybrid feature matching module takes the point-based feature descriptors extracted by the selected feature extraction module. These are compared against an internal history of previously observed feature descriptors. The comparison is carried out using the algorithms described previously (FLANN for SURF and SIFT and brute force L2 for BRISK). Once distance metrics have been computed for all the pairs currently observed and previously observed the confidence metric is calculated. This is done by an implementation of the algorithm described in Equations 3.1 - 3.3.

The depth perception module is an implementation of the direct depth algorithm described in Section 3.4. For each frame captured by the camera, it is passed a list of bounding boxes output by the selected saliency algorithm.

A full description of the structure of PM-SLAM can be found in Appendix B.

3.6 Experimental Results and Analysis

3.6.1 Performance Objectives

In order to evaluate the performance of the system, the following features needed to be demonstrated:

- **The system must be able to loop close.** Loop closure is the mechanism by which a SLAM system is able to reduce the error of its localisation estimate upon encountering previously observed environmental features. In order to assess whether a SLAM system is able to loop close, the rover’s trajectory must allow it to re-encounter features from early in its traverse in non-sequential images.

- **The end product must be an accurate position estimate.** A well-performing SLAM system should produce a better position estimate than the rover’s dead reckoning systems are able to on their own and produce results comparable to modern VO systems. The accuracy of the rover position is normally measured in comparison to a ground truth path, for example the path as measured by an on-board GPS receiver.
Chapter 3. Planetary Monocular Simultaneous Localisation And Mapping

Figure 3.16: Example images from: (a) PANGU simulated dataset, (b) WW dataset, (c) SEEKER Atacama desert dataset.

- **Low computation complexity** is a requirement of PM-SLAM, as the planetary hardware is often constrained.

- **A modular design** ensures the system can be used as a platform going forward to improve the functionality, performance and capabilities of PM-SLAM.

### 3.6.2 Validation Datasets

In order to evaluate the performance of PM-SLAM three sets of data are used. The first set is generated using the Planetary Asteroid Natural scene Generation Utility (PANGU). This tool is used to create a simulated planetary terrain, populated with randomly distributed boulders and craters. Having full control over this simulation
allows fine controlled placing of the rover and camera for capturing images of the surface, and it was used to capture a dataset which would evaluate the system’s ability to loop close. The dataset comprises a total 479 images (512 x 512 pixel monocular images captured from the rover position at each step of the traverse, Figure 3.16(a)), covering a total distance of 60.1 metres. In order to capture the image the camera position in the simulation at each step was calculated using the control signal summed with Gaussian noise.

The first of the non-simulated datasets was captured in West Wittering (WW), on the south coast of England collected by the STAR Lab. The selected site was a fairly level area of beach, chosen because the sand was a good analogue for the expected wheel slip a vehicle would experience on Martian soil. The test-site contained numerous dunes and rocks and larger artificial boulders were added, to ensure reasonable visual approximation of a Martian surface. The data was captured using the SMART robotic platform developed by the STAR lab (Figure 3.17a). The rover performed a 10.8 metre traverse, and the camera output was sampled at the rate of the control signal, which produced 172 images of 640 x 480 pixels (Figure 3.16(b)). The control signal used as input to PM-SLAM was the output of the wheel encoders on the rover.

The second real-world dataset was captured as part of an ESA field trial in the Atacama Desert, Chile in 2012 using the SEEKER rover platform (Figure 3.17b). The location was selected as an analog of a planetary surface, largely homogeneous with sparsely distributed boulders. A 102 metre traverse was chosen, which contained 2711 images which were 512 x 384 pixels (Figure 3.16(c)). Due to presence of a horizon in these images the top 80 lines of pixels were ignored. This ensured that the sky and the horizon would not be detected as salient features. In a real planetary mission the camera could be oriented to ensure no sky was visible in the images. If this was not possible, the number of lines making up the horizon could either be set manually as part of any calibration phase. There are also several horizon detection algorithms that could be used to automate this process [98, 99]. The control signal was recorded from the wheel encoders of the rover and were subsequently synchronised by timestamp with the images. The dataset also included DGPS (Differential Global Positioning System) tracking of the rover as well as the results of an on-board visual odometry.
Figure 3.17: (a) West Wittering trials utilised the SMART rover platform equipped with monocular cameras and DGPS for groundtruthing, (b) SEEKER rover performing trials in Chile (ESA). Both rovers demonstrate the high pitch angle of cameras mounted on planetary platforms.

(VO) localisation system, which is used here for comparative analysis. The VO was provided by the Roke Manor DROID structure-from-motion algorithm and the Oxford RSLAM [73] algorithm [106], which represent state of the art in VO for space systems. The independently optimised VO data was used as a comparison against the PM-SLAM system’s performance. Due to recording issues with the DGPS data some ground truth data is missing during random short intervals over the traverse. In order to utilise the data, it was sampled randomly throughout the traverse, instead of strictly comparing every single time step. As a result no average error is displayed, but the performance of PM-SLAM against the groundtruth and on-board localisation is demonstrated visually.

3.6.3 Results and analysis

All of the experiments were performed offline, on a desktop PC with 12GB RAM, 2.5GHz Intel Xeon 4-core processor running 64-bit Ubuntu 12.04 and ROS Hydro. The software was developed as a prototype and tested on hardware more capable than space qualified hardware, however the goal was to show that the hybrid-saliency system could act as a robust input to the SLAM filters. In order to validate that system could function on low specification hardware a single run was performed on the SEEKER
Table 3.3: PM-SLAM timings combined with other elements of rover GNC. Each process is listed, along with the interval at which it is carried out and the time taken each time the process is run. The final column indicates the average number of seconds required by each process for a 1cm traverse.

dataset, testing the performance over 143 steps (approximately 143 metres). For this test the system was limited to a single core capped at 250Mhz and 50MB of RAM. The traverse took 633.121 seconds to complete, with step times ranging from 1.009 to 10.031 seconds, with a mean of 4.427 seconds per step. Table 3.3 shows how this fits into a larger GNC framework to meet the speed requirements from Section 1.2. Each of the elements of a GNC system will be triggered at different intervals, in this case specified as the distance of traverse in centimetres between triggerings. At these intervals each process is performed and the time required is listed in seconds. From these we can compute the average amount of time that each of the processes contributes to 1 centimetre of travel. Summing these gives a total of 0.468 seconds required by all the processes per centimetre of travel. Taking the inverse of this gives an overall speed of 2.13cm/s. A cumulative graph of the step times in this experiment are shown in Figure 3.18.

Figure 3.19 shows a plot of the results from the system using an EKF as the SLAM filter, while the hybrid-salient feature tracking is performed using a combination of Hou saliency and SURF descriptors. The plot shows the comparison between the PM-SLAM estimated rover path, the path as given by the odometry (or control signal in this case) and the ground truth (the simulated path the rover traversed and recorded images from). The PM-SLAM rover position estimate is closer to the ground truth plot throughout the traverse, compared to the odometry (control) path. The system also shows a high accuracy once the rovers has completed a full loop and encounters objects
Chapter 3. Planetary Monocular Simultaneous Localisation And Mapping

Figure 3.18: PM-SLAM cumulative step times on limited hardware setup.

it has observed previously, which demonstrates loop closure - one of the key features of PM-SLAM’s system design.

The $L_2$ norm has been used for determining the error between the groundtruth position ($P_{truth}$) and the estimated ($P_{est}$) and planned position ($P_{plan}$) respectively,

$$
\varepsilon_{position}^{est} = \|P_{est} - P_{truth}\| \\
\varepsilon_{position}^{plan} = \|P_{plan} - P_{truth}\|
$$

(3.10)  
(3.11)

The heading estimate is less important than the position estimate for planetary exploration, however this can be augmented by the use of a sun sensor [97].

The time for a single step is given as the amount of time between the image and the control signal being provided to PM-SLAM, and the system producing a rover position estimate as an output. These step times were then averaged.

Figures 3.20 & 3.22 illustrate the absolute rover position error between the PM-SLAM position estimate and the ground truth rover position, averaged over an entire traverse. The dead reckoning is included for comparison as well. In the case of PANGU this is
3.6. Experimental Results and Analysis

Figure 3.19: PM-SLAM loop closure demonstrated using an EKF SLAM filter, with Hou-SURF hybrid-salient features using the PANGU dataset. Traverse is shown in metres.

The control data and for West Wittering, which is a real world dataset, this is the wheel odometry readings.

PM-SLAM was shown to have a greater average rover position accuracy than the dead reckoning using the simulated PANGU data, regardless of the combination of modules chosen (Figure 3.20). Different combinations of the modules gave varied accuracy, but on the whole they performed similarly over the different datasets. Varying the combinations of modules did affect the computation time, with Hou-SURF running the fastest and Rudinac-BRISK running the slowest. The PANGU dataset proved challenging in unexpected ways. PANGU datasets are generated using a set of 3-D boulder models which are then textured to appear like planetary rocks. This means that over a long traverse it is likely that boulders will be repeated, and similar textures will be applied. This leads to identical visual features being observed at different locations within the dataset, which was challenging for the point-based feature extraction algorithms. How-
Figure 3.20: PM-SLAM results for the PANGU dataset, showing (a) average absolute rover position error for PANGU and (b) the average per-step computation time using different combinations of hybrid-saliency and SLAM filters.
3.6. Experimental Results and Analysis

![PM-SLAM heading accuracy using PANGU accuracy]

**Figure 3.21**: PM-SLAM heading results for the PANGU dataset showing the absolute rover heading error in radians for each of the combinations of

ever the confidence metric used in the hybrid-saliency worked well to compensate for this issue, and allowed the hybrid-saliency system to remove outliers.

The real world WW dataset provided the system with extremely noisy wheel odometry data, due to the large amount of wheel-slip and the skid steering dynamics of the rover platform. This is visible in the dead-reckoning error seen in Figure 3.22. By tuning the various elements of PM-SLAM, including the hybrid-saliency tracking (i.e. the number of SURF features and the confidence metrics in Equation 3.3) and the SLAM filter confidences, a good position estimate was achieved by the system, giving much more accurate results relative to the dead reckoning system alone (Figure 3.22).

The different combinations of modules demonstrated similar consistency in terms of accuracy compared to the PANGU data (Figure 3.20) and Hou-SURF and Rudinac-Brisk were shown to again have the fastest and slowest step-computation times (Figure 3.22).

The BRISK feature detection system was shown to have the longest average compu-
Chapter 3. Planetary Monocular Simultaneous Localisation And Mapping

PM-SLAM accuracy using West Wittering dataset

Figure 3.22: PM-SLAM results for the WW dataset, showing (a) average absolute rover position error for WW and (b) the average per-step computation time using different combinations of hybrid-saliency and SLAM filters.
3.6 Experimental Results and Analysis

The difference in performance highlights the benefits of the modularity of the system, which allows different algorithms to be plugged in to a common architecture. For example, on average hybrid-salient features using SIFT detection tended to outperform other module combinations (Figures 3.20 & 3.22), however the SURF algorithm was markedly faster (Figures 3.20 & 3.22). Selection of an optimal set of modules would depend on the specific scenario and hardware constraints of the system as a whole, and the reconfigurable nature of the modular system allows for this.

The final test involved the SEEKER dataset. The most accurate configuration of PM-SLAM, determined from the earlier results (Figures 3.20 & 3.22) was selected to test with this dataset. This configuration used the EKF SLAM filter, together with Hou-
Figure 3.24: Absolute rover position error for SEEKER dataset over a 102 metre traverse, showing dead-reckoning, on-board localisation (VO) and PM-SLAM results.

SURF hybrid-salient features. Figure 3.24 shows the rover position error over a long linear traverse of the SEEKER rover. All of the localisation methods show drift, in that the localisation error tends to increase over time. This is due to the observability of pose estimation using odometry and landmark-based measurements. It has been shown that in landmark SLAM the state vector is rank deficient [79]. While the on-board VO system performs favourably compared to the dead-reckoning (provided by the wheel encoders), over the whole traverse, PM-SLAM’s localisation estimate was more accurate. The average step time for this traverse was 0.641 seconds for PM-SLAM.

Figures 3.25 and 3.26 show the 3-sigma error bounds of the filter over the SEEKER traverse. The y and heading error bounds are stable, whereas the x-direction shows a monotonic increase. This indicated that the estimation of the x direction is not stable, and that prediction error could grow over the course of the traverse. This may be due to the fact that the traverse in SEEKER dataset is a straight line traverse. This offers opportunity for objects in the y-direction to be observed moving over several frames, with minimal change in their position, however the there was very minimal lateral movement, which did not allow as much frame to frame loop closure. Further testing with more varied traverse shapes would help to investigate this further.

The implementation of PM-SLAM using the ROS platform allowed for easy development of modules. Figure 3.27 demonstrates the ability to easily change between config-
Figure 3.25: PM-SLAM rover position errors for the (a) x (lateral) traverse and the (b) y (forward) traverse components. The dotted lines represent the $3\sigma$ error bounds of the filter.
3.7 Conclusions and future work

The performance objectives outlined in Section 3.6.1 were achieved. The PANGU dataset results (Figure 3.19) clearly demonstrate PM-SLAM’s ability to loop close as well as good level of accuracy and fast computation (Figure 3.20). Figures 3.22 & 3.24 both show that the system provides much better accuracy than relying on dead-
3.7. Conclusions and future work

Figure 3.27: ROS node diagrams showing live examples of PM-SLAM nodes and services. (a) PM-SLAM using an EKF filter with Hou-SURF hybrid-saliency feature detection and tracking. (b) PM-SLAM using an EIF filter with Rudinac-SIFT hybrid-saliency feature detection and tracking.
reckoning alone when using challenging real world datasets. Furthermore, Figure 3.24 illustrates that PM-SLAM can perform favourably against an independently optimised state-of-the-art VO solution, which has been trialled for similar applications, namely planetary exploration. The low computational complexity was evident in the step-computation times in Figures 3.20 & 3.22, as well as the SEEKER dataset average step-computation time. The implemented modular design enabled the use and comparison of several combinations of PM-SLAM elements, and provides and extensible platform for future development.

PM-SLAM’s novel feature detection, tracking and depth perception systems, namely hybrid-salient features in combination with direct depth, has been shown to provide a lightweight, robust and extensible input to a planetary SLAM system. When tested on real world and simulated datasets, the system performed well demonstrating that the assumptions made in the design were sufficient and well founded for the scenario.

PM-SLAM modular design can be used to implement and perform comparative analysis on further SLAM filter and feature detection techniques. These could easily be added as a component to the system, and its performance could be evaluated and compared against the current PM-SLAM modules.

Due to the limited availability of datasets for testing visual navigation for planetary exploration, any future work would need to include the collection of a large dataset, with long traverses through suitable environments with adequate groundtruthing for vehicle position, and traverses that include circuits for loop closure evaluation.
4.1 Introduction

The existing methods for autonomous planetary rover localisation (described in Section 2.1), provide relative localisation. This means that the rover localises itself based on its previous position estimate.

In the case of VO based systems, each step of the localisation estimate calculates the change in position from the previous step and then adds this to the previous localisation estimate. This means that the error in each step is propagated to the next step, leading to an accumulation of error and the localisation estimate drifting over time.

For SLAM-based systems the localisation is partially based on the previous step localisation estimate, as well as a map of previously encountered features saved over the course of the rover’s traverse. When features are added to the map, their location is determined according the estimated position of the rover at the time that the feature is encountered for the first time. This means that the position of features in the map and the localisation estimate have an error dependent on the accuracy of the previous
Chapter 4. Orbiter Localisation

step’s localisation. SLAM algorithms incorporate a probabilistic quantification of the expected error in both rover position and map feature location, and re-encountering map features helps to minimise these errors and correct the drift in localisation estimate from the true rover position. However, in the case of a trajectory without a loop closure opportunity, the localisation estimate from a SLAM system will experience drift.

In currently used techniques described in Section 2.1, this drift is constrained by a combination of limiting the traverse distances per sol and also using human intervention from the groundstation to accurately localise the rover manually, before starting the next day’s traverse. This manual localisation uses the estimated position of the rover, together with in-situ images of the surface encountered by the rover to locate the rover’s position in a global map of the planet. This global (or absolute) localisation gives the rover position in terms of the cartographic co-ordinate system of the planet.

Section 2.4 outlined methods being investigated for automating this process of global map localisation. These methods focussed on comparing the arrangement of features in the global map (the absolute map from orbital images) with the local map (the relative map from the rover’s measurements and localisation estimate). In this chapter an alternative method is investigated, which also uses the absolute map to improve localisation accuracy.

The orbital mask matching algorithm described in this chapter builds upon the system developed in Chapter 3. Information about feature positions in the global map are used to select features for use in the rover’s visual feature based localisation system. A proof of concept design is described along with some initial tests performed on a simulated dataset.

4.2 System Overview

The system uses models of large features contained in the global map to store feature descriptors. This enables matching of local features in the rover’s visual scene with the global map.

Figure 4.1 describes the steps used to localise the rover. The rover initialises at a
4.2. System Overview

Figure 4.1: System overview, for associating point based feature descriptors with global map features for rover localisation.

known location, determined by pinpoint landing or ground station based localisation. A model is used to determine the shape of the rover in the local frame of the rover, using a cylinder shape (Section 4.4.1) or a dome shape (Section 4.4.2). The process for developing these models is described in Section 4.4. An image mask can be computed using a modified version of the direct depth algorithm described in Section 3.4. The modifications are described in Section 4.3.

The feature points detected in the scene are matched with the feature points detected in previous steps, which have been associated with the global map features. Using the height of the features stored in the global map together with the direct depth with elevation algorithm in Section 4.3 a local point cloud is created. A corresponding point cloud can be extracted from the features stored in the global map. Section 4.5.3 describes a method for iteratively determining the rigid transform between the two point clouds. The transform parameters can then be used to determine the position of the rover in the global map.

With the rover position in the global map known, the masks can be used to associate locally features with orbital map features. The features can then be projected onto the model as described in Sections 4.5.1 and 4.5.2. The 3D projections of the local points can then be added to the boulders in the global map for use in future stages.
4.3 Elevation and Inverse Direct Depth

The algorithm described in this section uses a modified version of the direct depth algorithm described in 3.4. In the previously discussed implementation, features detected in the image plane were projected to the flat ground of the local real-world co-ordinate frame of the rover.

If the elevation of the feature from the ground is known, then direct depth can determine the real-world x and y co-ordinates by projecting to a flat plane at the known height of the object, as shown in Figure 4.2.

In order to calculate the direct depth of point $P'$, which has real world co-ordinates $(x', y', e)$ we modify the algorithm used to determine $P$, which had co-ordinates $(x, y, 0)$. Equation 4.1 shows the modification to Equations 3.6, 3.7, 3.8 and 3.7 to include the
4.4. Orbiter Masking

elevation, \( e \).

\[
\tan(\beta) = \left( \frac{2v}{V} - 1 \right) \tan\left( \frac{V \tan(\frac{VFOV}{2})}{2} \right)
\]
\[
y' = \frac{h - e}{\tan(\alpha + \beta)}
\]
\[
\tan(\gamma') = \frac{(u - \frac{U}{2})}{\sqrt{\left(\frac{U}{2} \tan(\frac{HFOV}{2})\right)^2 + v^2}}
\]
\[
x' = (\sqrt{(h - e)^2 + y'^2}) \tan(\gamma')
\]

In Section 3.4, the direct depth algorithm was used to map features from the image plane to the real-world co-ordinates. Inverse direct depth performs the inverse of this process, by mapping the real-world co-ordinates of features \((x, y, z)\) to the image plane co-ordinates \((u, v)\) using the camera parameters (the camera height \( h \), the vertical and horizontal field of view \( HFOV \) and \( VFOV \), and the image width and height \( U \) and \( V \) as shown in Equation (4.2)).

\[
v = \frac{V}{2} + \left( \frac{\frac{V}{2} \tan(\frac{VFOV}{2}) \tan(\frac{\pi}{2} - \alpha - \tan^{-1}(\frac{y}{H}))}{\sqrt{\frac{V}{2} \tan(\frac{HFOV}{2})^2 + y^2}} \right)
\]
\[
u = \frac{U}{2} + \left( \frac{x}{\sqrt{H^2 + y^2}} \right) \left( \frac{\frac{U}{2} \tan(\frac{HFOV}{2})^2 + y^2}{2} \right)
\]

4.4 Orbiter Masking

The orbiter mask matching algorithm is a proof of concept method for allowing tracking of orbital features, in the global map, using descriptors generated from sensors on the surface. This allows the rover’s local relative map and the orbital absolute map to be co-registered, and can help the rover to localise on the surface.

The orbital map data is in the form of a list of two dimensional co-ordinates, representing the the locations of large features, i.e. boulders on the planetary surface. Due to the limitations of resolution achievable using cameras mounted on board observation satellites only features with diameters larger than 0.5 metres are likely to be detected \([71], [41]\). Furthermore the height of the features, and the elevation of the terrain
at that point cannot be accurately determined from the orbital image, except in the case of significant slopes. In addition, no further information except for the centroid of the boulder is included, as size and shape estimates are likely to be highly inaccurate.

In the work described the boulder positions are only used to aid in localisation of a planetary vehicle. In order to perform hazard avoidance, more traditional methods would need to be implemented on the rover platform as boulders that would be harmful to the rover could be potentially smaller than those detected from orbital imagery. In addition other hazards such as crevasses or dangerous steep would not be detectable using the orbital map.

There are several methods available for determining the location of features from orbital maps, including automated and manual methods. Some of these methods were discussed in Section 4.5. Very simple methods such as applying a difference of Gaussians can extract the high frequency elements from the relatively flat landscape which can then be thresholded using Otsu’s method. An example of this technique is shown in Figure 4.3, in which a difference of Gaussian and then Otsu binarisation was performed on a simulated orbital image generated using the PANGU tool.

In the context of a planetary exploration mission, it is assumed that the orbital map will be generated offline, well in advance of launch of the mission, and would be pre-loaded onto the rover. Therefore, regardless of whether the map has been generated manually, or using an automated image processing technique, the map is assumed to have been verified and have a high level of accuracy, i.e. there would be no false positive identifications of features.

In order to generate descriptors for the features in the orbital map, the initial well-known localised position of the rover is used, generated by pinpoint landing data or by initial human localisation after the rover has landed. Given the initial location of the rover in the global map the system can now determine the rocks that should be visible in the rover’s field of view. This determined by utilising knowledge of the rover’s camera position and parameters to create a region of interest which should contain all the boulders visible to the rover. The bounds of this region are calculated using Equation 4.3, and the an example of the results are shown in Figure 4.4. The features extracted
Figure 4.3: Example of extracting features from orbital image using difference of Gaussian and Otsu Thresholding. (a) An example orbiter image generated using PANGU. (b) Blob map generated after applying difference of Gaussians to the orbiter image. (c) Binarised map generated using Otsu’s method on blob map.
from the map are assumed to have a 1 metre diameter, so a circle can be drawn onto the map representing the extent of the boulder on the map, even though the global map represents boulders only as 2-dimensional co-ordinates. Any of these circles that intersect with the rover’s field of view should be visible in the images taken on the planetary surface.

We can determine the location of large boulders in the rover’s field of view by applying the inverse of the direct depth equations described in Section 3.4. First the features are transformed to the rover’s local frame using Equation 4.4. Then inverse direct depth can be performed using Equation 4.2. The centroids of the features are assumed to lie flat on the surface, and therefore have $z$ co-ordinates of 0.

$$\begin{align*}
x_A &= r_x, y_A = r_y \\
x_B &= x_A - R \sin \left( \frac{HFOV}{2} \right), y_B = y_A + \cos \left( \frac{HFOV}{2} \right) \\
x_C &= x_A + R \sin \left( \frac{HFOV}{2} \right), y_C = y_B
\end{align*} \quad (4.3)$$

We can determine the location of large boulders in the rover’s field of view by applying the inverse of the direct depth equations described in Section 3.4. First the features are transformed to the rover’s local frame using Equation 4.4. Then inverse direct depth can be performed using Equation 4.2. The centroids of the features are assumed to lie flat on the surface, and therefore have $z$ co-ordinates of 0.

$$\begin{bmatrix} x_{f,\text{local}} \\ y_{f,\text{local}} \end{bmatrix} = \begin{bmatrix} x_{f,\text{global}} \\ y_{f,\text{global}} \end{bmatrix} - \begin{bmatrix} x_{\text{rover}} \\ y_{\text{rover}} \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta_{\text{rover}}) & \sin(\theta_{\text{rover}}) \\ -\sin(\theta_{\text{rover}}) & \cos(\theta_{\text{rover}}) \end{bmatrix} \quad (4.4)$$

Where $(x_{\text{rover}}, y_{\text{rover}}, \theta_{\text{rover}})$ is the rover position in the global map frame, and $(x_{f,\text{global}}, y_{f,\text{global}})$ is the rover feature position in the global map frame.

Performing this calculation we can determine the position of boulders in the image plane as shown in figure 4.5.

Due to the limited nature of the orbital data, the centroid of the boulders and the fact that the widths are greater than 1m (due to the orbital image resolution) are the only pieces of information that can be extracted. Having used the centroid co-ordinates in order to determine the image plane location of the boulder centroid, we can also use assumptions about the size to develop a model for a boulder on the planetary surface.

If we make an initial assumption that the boulders are 1 metre in diameter, we can take the boulder footprint from the orbital map and draw it onto the image plane. The
4.4. Orbiter Masking

Figure 4.4: (a) Top down view of boulders on planetary surface. (b) Example of orbiter map, where boulders are stored as points. (c) The rover position is shown as a pink circle. The orbiter map can be used to determine the positions of boulders in the area surrounding the rover. These are shown as green circles, with their size representing the 1m minimum diameter of features visible from orbit. The heading of the rover along with knowledge of the field of view (bounded by the blue lines) allows determination of which boulders should be visible to the rover’s camera.
Figure 4.5: (a) The global map, with boulder positions shown as red crosses. The rover is located at the pink circle and has a field of view bounded by the blue lines. (b) Local map, where the boulders outside of the field of view have been removed and the map transformed to the rover’s local perspective. (c) Image showing boulder centroids in the image plane, calculated using the local map and the inverse direct depth algorithm.
4.4. Orbiter Masking

Figure 4.6: (a) The footprint of each boulder if assumed to be 1 metre in diameter. (b) The boulder footprints transformed to the image plane using inverse direct depth.

Having determined the footprints of the boulders in the visual scene, a model shape can be used to create an image mask for the rover. In the examples discussed in this chapter, two models are used; a cylinder and a dome. The cylinder model is selected as an enclosing shape based on [36] where as the use of a dome is a novel model, introduced in this work in order to attempt to better approximate the shape of boulders on the surface.

4.4.1 Cylinder Model

The cylinder boulder model assumes that the boulders can be represented by upright cylinders, with the footprints demonstrated in Figure 4.6 as their bases. The top and
The bottom of the circles are defined by the circles given in Equation 4.5

\[
x_{bottom} = x_{top} = x \\
y_{bottom} = y_{top} = y \\
z_{bottom} = 0 \\
z_{top} = e \\
(x - x_c)^2 + (y - y_c)^2 = r^2
\] (4.5)

Where \((x_c, y_c)\) is the location of the boulder centroid in the local map, \(r\) is the boulder radius and \(e\) is the height, set to 0 for the base and the height of the cylinder model for the top.

Using inverse direct depth the points for the top and the base of the cylinders representing each boulder in the scene can be transformed to the image plane. The maximum and minimum x-axis pixels from the circles representing the base and the top of the cylinder are used as the corner points of the quadrilateral representing the visible face of the cylinder. An example of the calculated cylinder outlines is shown in Figure 4.7 and the image mask generated from the cylinders is shown in 4.8.

### 4.4.2 Dome Model

The calculation of the domed model to represent the boulders is more complex than the cylindrical model described previously. Again the boulder footprint shown in Figure 4.6a is used as the base of the dome. The dome itself is represented by a semi-ellipsoid with equal minor axes (giving it a circular base, as per the footprint). The plane formed by the two minor axes forms the base of the dome and is adjacent to the ground plane. The equation for the points of an ellipsoid are shown in Equation 4.6

\[
\frac{(x^2 - x_c^2)}{a^2} + \frac{(y^2 - y_c^2)}{b^2} + \frac{(z^2 - z_c^2)}{c^2} = 1
\] (4.6)

Where \(a\) and \(b\) represent the semi-minor axes and are equivalent to the radius of the boulder and \(c\) represents the semi-major axis and is equivalent to the boulder height.
4.4. Orbiter Masking

Figure 4.7: Outline of cylinders representing boulder features mapped onto the image plane using inverse direct depth. The pink circle represents the projected centroid from the map.

Figure 4.8: The image mask generated from the cylinder models.
Chapter 4. Orbiter Localisation

The aspect of the dome visible in the image plane is a semi-elliptical cross section, parallel to the image plane. The minor axis length of this cross section is equal to the diameter of the footprint, as this is the base of the dome. In order to draw the cross section in the image plane, the length of the semi major axis of the cross section must be calculated. Given that the image plane is orthogonal to the tilt angle of the camera ($\alpha$), the angle of the cross section in the z-y plane ($\phi$) can be easily calculated. This information can be used to determine the length of the semi major axis of the cross section.

Figure 4.9 shows the parameters used to calculate the dome cross section semi-major axis $q$. In the case of a 1 metre diameter boulder model, $a = 0.5$, and $b$ is the modelled height of the boulder. The dome cross section is seen as a line running from the bottom front of the boulder ($q_1 = (-a, 0)$) and the point where the cross section intersects the top of the dome ($q_2 = (p, n)$). For the purposes of the calculation the centroid of the
dome (and therefore the boulder) is translated to the origin.

\[\phi = 90^\circ - \alpha\]

\[n = m \tan(\phi) = (a + p) \tan(\phi)\]  

\[\frac{p^2}{a^2} + \frac{n^2}{c^2} = 1\]  

(4.7)

The equation for the line representing the cross section, in terms of \(p\) and \(n\) is used together with the equation for the dome ellipse in profile. By substituting one into the other, a quadratic equation is found in terms of \(p\). The two roots of this equation represent the two points where the cross section intersects the profile of the dome, at the bottom and near to the top. The value not representing the bottom point is used to calculate \(m\) and using Pythagoras’ theorem the length of \(q\) is computed.

Using the calculated value of \(q\), the positions of the points of the cross section in 3-dimensional space can be calculated, again using the 2-dimensional ellipse equations.

\[z = \frac{ab}{\sqrt{a^2 + b^2 \tan^2(\psi)}} \cdot \sin(\phi)\]

\[y = \frac{z}{b} \cdot (p + a) + y_c\]  

\[x = z \cdot \tan(\psi) + x_c\]  

(4.8)

In these equations \(\psi\) represents the deflection angle for each point around the ellipse. The co-ordinate \((x_c, y_c)\) represents the centroid of the boulder. Once the real world points for the cross section ellipse have been determined, inverse direct depth can be used to transform them to the image plane allowing the domes to be drawn onto the rover camera images as demonstrated in Figure 4.10. An example of the image mask extracted from these outlines is shown in Figure 4.11.

### 4.4.3 Mask Occlusions

In certain cases, where boulders are positioned in front of other boulders, the masks generated for each may overlap. In order to ensure any point features detected are associated with the correct boulder the masks need to be modified.
Chapter 4. Orbiter Localisation

Figure 4.10: Dome representations of boulders drawn onto the image plane.

Figure 4.11: The image mask generated from the dome models.
When two features’ masks overlap, it is assumed that the mask of the feature nearer to the camera (i.e. the feature further in front) should include the overlapping area, and this area should be removed from the other feature’s mask.

In Figure 4.12 the feature masks for features 3 and 4 overlap, as marked by the red area. The orbiter map is used to determine that feature 4 is positioned closer to the rover than feature 3. Therefore the shared portion of the masks is included in the mask for feature 4 (Figure 4.12b) and removed from the mask for feature 3 as shown in (Figure 4.12c).

4.5 Mask Matching

The boulder masks created using the method in Section 4.4 can be used for feature matching and tracking in a similar manner to the bounding boxes used in the hybrid-saliency system used in PM-SLAM (Section 3.3). This works well when the position of the rover in the global map is known, as shown in Figure 4.13.

In the case of rover localisation for a rover using only dead reckoning or similar systems, the rover position will not be well known and hence, the rover position estimate would have to be refined before creating the orbital feature masks at each step of the traverse.

To achieve this, instead of matching the hybrid features in the previous steps with hybrid features in the current step, point based features detected across the whole of the current image, with no mask applied, are matched with the history of hybrid features in order to determine which features are present in the current image.

Once a feature has been matched with its counterpart in the global map the position of the centroid of the feature needs to be calculated in order to use the information to aid in localisation. In order to do this the point based features detected for each boulder are projected on to the model shape used to generate the masks, either the cylinder model (Section 4.5.1) or the dome model (Section 4.5.2).
Figure 4.12:  
(a) Feature masks, with overlapping areas highlighted in red.  
(b) Feature mask for feature 4 including overlap area.  
(c) Feature mask for feature 3 excluding overlap area.
4.5. Mask Matching

![Figure 4.13: Example of matching features using orbiter masks with hybrid-saliency.](a) (b)

4.5.1 Projecting Points onto a Cylinder

In order to project point based features detected in the image plane on to the cylinder model representing the boulder, we solve the parametric equations for the ray line running from the camera through the feature point in the image plane down to the ground, and the equation for a cylinder of the height and radius used in the model.

The equation for the ray line can be calculated by selecting two points along its length. For simplicity we select the point on the ground (0 metres from the ground), and the point halfway between the camera and the top of the cylinder. The real world co-ordinates of these two points ($P_1$ and $P_2$ in Figure 4.14) can be calculated using the modified direct depth equations in Section 4.3, using these two heights as the elevation ($e$). The equation for any point on the ray in terms of the co-ordinates of these two points is given in Equation 4.9.
In the cylinder boulder model the equation defining the bounds of the model is represented by the edges of the footprint circle at every height up to the height of the cylinder, and above this height, any point within the footprint circle. In Figure 4.15 a cylinder centred at \((x_c, y_c)\) is viewed from a top-down view and a ray is shown intersecting the cylinder twice. The first step of mapping a point-based feature to the cylinder is to solve the parametric equation for the cylinder and the line. The equation of the cylinder is given in Equation 4.10

\[(x - x_c)^2 + (y - y_c)^2 = r^2 \tag{4.10}\]

Solving these two equations gives the quadratic in terms of \(d\) (shown in Equation 4.11), which represents the points where the ray line intersects the cylinder, using Equation

\[P_1 = (x_1, y_1, z_1)\]
\[P_2 = (x_2, y_2, z_2)\]

\[x = x_1 + d(x_2 - x_1)\]
\[y = y_1 + d(y_2 - y_1)\]
\[z = z_1 + d(z_2 - z_1)\]
4.5. Mask Matching

Figure 4.15: Top down view of a cylinder model for a feature, within the rover field of view including a ray through a point feature which intersects the cylinder.

These two points are shown as red dots in Figure 4.15.

\[
(x_1 + d(x_2 - x_1) - x_c)^2 + (y_1 + d(y_2 - y_1) - y_c)^2 = r^2
\]
\[
d^2 \left[ \frac{(x_2 - x_1)^2 + (y_2 - y_1)^2}{r^2} \right] - 2d \left[ \frac{(x_1 - x_2) \cdot (x_1 - x_c) + (y_1 - y_2) \cdot (y_1 - y_c)}{r^2} \right] + \frac{(x_1 - x_c) + (y_1 - y_c)}{r^2} = 0
\]

(4.11)

The root which gives an intersection closer to the rover (i.e. with a smaller real world y co-ordinate) represents where the line intersects the cylinder in front of the rover, whereas the other root and corresponding point represent the point on the back of the cylinder, not visible to the rover.

If the real world z value calculated for the intersection is larger than the height of the cylinder model, then the point is assumed to lie on the flat top end of the cylinder. The real world point can then be calculated by applying the direct depth algorithm, with
Chapter 4. Orbiter Localisation

4.5.2 Projecting Points onto a Dome

The method for projecting detected feature points onto the dome model is similar to the method used for projecting onto the cylinder model. Again, the intersections of the ray from the camera through the feature point (as described in Equation 4.9) in the image plane with the boulder model is used to determine its 3-dimensional real world position.

The points on the surface of the semi-ovoid shape representing the dome are represented by Equation 4.6. The equation for the ray and the ellipsoid are combined in to give a
quadratic in $d$ as shown in Equation 4.12

$$\frac{(x_1 + d(x_2 - x_1) - x_c)^2}{a^2} + \frac{(y_1 + d(y_2 - y_1) - y_c)^2}{b^2} + \frac{(z_1 + d(z_2 - z_1) - z_c)^2}{c^2} = 1$$

$$a = b$$

$$d^2 \left[ \frac{(x_2 - x_1)^2 + (y_2 - y_1)^2}{a^2} + \frac{(z_2 - z_1)^2}{c^2} \right] - 2d \left[ \frac{(x_1 - x_2) \cdot (x_1 - x_c) + (y_1 - y_2) \cdot (y_1 - y_c)}{a^2} + \frac{(z_1 - z_2) \cdot (z_1 - z_c)}{c^2} \right] + \frac{(x_1 - x_c) + (y_1 - y_c) + (z_1 - z_c)}{a^2} + \frac{(z_1 - z_c)}{c^2} = 0$$

(4.12)

Solving this quadratic gives two roots, which represent the two points where the ray intersects the dome. One of these points is towards the front of the boulder, from the rover’s perspective and the other is on its rear face. The root that gives a point on the face visible to the rover is selected as the projected point position.

Figure 4.17 shows an example of projecting point based features from the image onto the real world co-ordinates of the cylinder. In the example evenly spaced points on the image were used instead of real feature points to demonstrate the projected shape.

### 4.5.3 Point Cloud Matching and Transformation

The rover starts from a well known location, and identifies boulders in its vicinity and attaches descriptors to boulder models representing them. As the rover traverses the planetary surface, it uses previously identified point features to help it localise using a process described in this section. Once the rover has used this information to improve its localisation estimate, it can then add further feature descriptors to its global map for use in subsequent steps.

The rover uses the estimated error of its own localisation estimate in order to define the area likely to contain its field of view. Figure 4.18 shows an example of the area containing the rover’s field of view. The bounding polygon of this area is defined in Equation 4.13, using the rover position $(x, y, \theta)$, the errors associated with each of these
parameters \((e_x, e_y, \epsilon_\theta)\), the rover’s horizontal field of view \(FOV\) and the range where useful features can be detected \(R\).

\[
\begin{align*}
A_x &= r_x - \sqrt{e_x^2 + e_y^2} \cdot \sin \left( \tan^{-1} \left( \frac{e_y}{e_x} \right) - \epsilon_\theta \right), \quad A_y = r_y - \sqrt{e_x^2 + e_y^2} \cdot \cos \left( \tan^{-1} \left( \frac{e_y}{e_x} \right) - \epsilon_\theta \right) \\
B_x &= r_x + \sqrt{e_x^2 + e_y^2} \cdot \cos \left( \tan^{-1} \left( \frac{e_y}{e_x} \right) - \epsilon_\theta \right), \quad B_y = r_y - \sqrt{e_x^2 + e_y^2} \cdot \sin \left( \tan^{-1} \left( \frac{e_y}{e_x} \right) - \epsilon_\theta \right) \\
C_x &= B_x + R \cdot \sin \left( \frac{FOV}{2} + \epsilon_\theta - \epsilon_\theta \right), \quad C_y = B_y + R \cdot \cos \left( \frac{FOV}{2} + \epsilon_\theta - \epsilon_\theta \right) \\
D_x &= r_x + R \cdot \sin \left( \frac{FOV}{2} - \epsilon_\theta \right), \quad D_y = r_y + R \cos \left( \frac{FOV}{2} - \epsilon_\theta \right) \\
E_x &= r_x - R \cdot \sin \left( \frac{FOV}{2} + \epsilon_\theta \right), \quad E_y = r_y + R \cos \left( \frac{FOV}{2} + \epsilon_\theta \right) \\
F_x &= A_x - R \cdot \sin \left( \frac{FOV}{2} + \epsilon_\theta + \epsilon_\theta \right), \quad F_y = A_y + R \cdot \cos \left( \frac{FOV}{2} + \epsilon_\theta + \epsilon_\theta \right)
\end{align*}
\]

(4.13)

When the rover moves and encounters a new visual scene, it detects point based features over the whole of the image. The descriptors of these point based features are matched against the database of point based feature descriptors associated with the boulders in the global map which lie in the area bounded in Figure 4.18. Limiting the search to the
Figure 4.18: The search space within the global map for feature points. The pink circle represents the rover position and the blue line, the bounds of the field of view including errors. The bounded area acts as a form of outlier rejection, ensuring matches are only found in an area that the rover is likely to be positioned. The sparse and homogeneous nature of planetary terrain means similar features can be encountered over the course of a long traverse.

When a match is found between a point in the rover’s visual field and the global map, it is assumed that the local feature has the same height as the height of the feature stored in the global feature map. This is the height previously calculated, when the feature was first detected and projected onto the boulder’s model as in Sections 4.5.1 and 4.5.2. The feature is projected into 3D space using direct depth (Section 4.3) with elevation set to this height. The resulting point cloud is shown in Figure 4.19a along with the global map point cloud generated from the database of descriptors from previous steps in Figure 4.19b.

There are now two point clouds, one representing the point-based features that the
Chapter 4. Orbiter Localisation

Figure 4.19: (a) Point cloud representing the point based features detected in the rovers local field, projected to the height of their matching feature in the global map. (b) Point cloud representing the corresponding feature points in the global map.

The rover has detected in the current step, and one representing the features in the global rover map. The next step is to determine the rigid transform that moves the local points to the global points, and hence discover the rover’s position in the global map.

The two point clouds are made up of points $P_{local,i}$ and $P_{global,i}$ where $i = 1, 2, ..., N$ for $N$ matches. The points are related by the rigid transform $P_{global,i} = R \cdot P_{local,i} + T + N_i$, where $R$ represents a rotation and $T$ represents a translation. Due to the fact that points are detected from visual scenes and projected on to an imperfect boulder model, some noise is incorporated ($N_i$), to account for small errors.

In order to determine the rigid transform between the point sets, a Singular Value Decomposition (SVD) method [3], [61] can be used. This method is commonly used as an element of ICP methods [83], [27].

First the centroids of both point clouds, $q_{local}$ and $q_{global}$, are calculated using the normalised mean of their respective points. The distance between each point in a point cloud, and the centroid of its point cloud are calculated, $d_{local}$ and $d_{global}$. These are then combined into a matrix $H$ and a SVD is performed, as in Equation 4.14.

\[
H = d_{local} \cdot d_{global}^T;
\]

\[
[U, \Sigma, V^*] = SVD(H);
\]  

(4.14)
4.5. Mask Matching

The rotation parameter of the rigid transform can then be determined using Equation 4.15.

\[
R = \begin{cases} 
V^* \cdot U^T & \text{if } \det(V^* U^T) = 1 \\
V^T \cdot U^T & \text{if } \det(V^* U^T) = -1 
\end{cases}
\]  

(4.15)

Once the rotation has been determined, the translation, \(T\), parameter of the rigid transform can be calculated using the point cloud centroids and the rotation matrix. The local centroid (\(q_{local}\)) is rotated using the rotation matrix (\(R\)). The distance between this new point and the centroid of the global point cloud (\(q_{global}\)) represents the translation, as shown in Equation 4.16.

\[
T = q_{global} - R \cdot q_{local}
\]

(4.16)

An example of results of using this transformation is shown in Figure 4.20.

The transformation produced will not be perfect, due to possible mismatching of point-based features and the projection of the points onto the imperfect boulder models. Although the matching is limited to boulders thought to be within the rovers FOV, mismatches can still occur within the field of view. These two factors contribute to the noise parameter \(N_i\) associated with the transform of each point. The transformation error due to this noise can be reduced iteratively. First the parameters of the rigid transform are calculated, using the SVD method. These parameters are used to transform the local point cloud, essentially placing them in the global map frame. It is then possible to calculate the euclidean distance between global point cloud, and the transformed local point cloud. Any pairs of points which have an error above a preset threshold are removed from both point clouds, and then the transformation parameters are recalculated. This process can be repeated until there are no pairs of points with errors above a certain threshold, or a maximum iteration limit is reached. This helps to remove outliers and improve the position estimate. Figure 4.21a shows the resulting point cloud (in green), when the local points (in blue) are transformed using the initial computation of the transform. The distance errors between each point in the transformed local point cloud (in green) and the points in the global point cloud (in red)
Figure 4.20: The local point cloud, shown in blue, has been transformed, resulting in the green points. The target points of the global point cloud are shown in red.

are shown in Figure 4.21b. The mean error is represented by the green line, and an example threshold has been set at 0.5 metres, represented by the red line. Figure 4.21c shows the second iteration, with the features with error greater than 0.5 metres in the previous iteration removed from the point clouds before calculating the rigid transform. As well as removing the largest error values from the previous iteration, Figure 4.21d shows that the overall mean error is also reduced, and therefore the accuracy transform parameters can be considered more accurate.

The rover is positioned at the origin of the local map. The position of the rover in the global map can be found using the rigid transform computed using the feature point clouds.

Once the rover position in the global map has been determined, the image can be masked as demonstrated in Section 4.4. Any features that have been newly detected can have their descriptors added to the boulder models in the global map for use in
In order to test the algorithm a simulated data set was constructed using PANGU. A boulderfield 250 x 200 metres was created, with large boulders likely to be visible in orbital imagery. A 190 metre straight traverse through the field was calculated, with 190 steps each of 1 metre plus Gaussian noise. An example image from the traverse are shown in Figure 4.22.

The orbital map was represented by the list of 2-dimensional cartesian co-ordinates of boulders used to generate the simulation. This list was filtered to only allow boulders
with a diameter over 1 metre. The boulder field and the text traverse are shown in Figure 4.23.

The traverse was run with both the cylinder and the dome boulder models in order to compare the performance of the two types of model.

The cylinder model seems to have better performance overall, but worse performance in the x-axis. This is likely due to the fact that the projection onto the straight face cylinder is more consistent from frame to frame, as compared to the dome, where slight changes in angle for the projection point can have a significant effect on the features 3-dimensional position. The worse performance in the x-axis is likely due to the fact that features higher up the cylinder have a greater error compared to their true position than the dome model which curves towards the feature as its height increases.

As the error increases significantly the models no longer cover the correct boulders. At this point, the system is operating as a conventional monocular VO system, and the points added to the global map will be attributed to the wrong features.

The results are comparable with other SURF based VO systems such as [53]. If the method used was combined with IMU data and incorporated into a SLAM architecture...
4.6. Experimental Results

Figure 4.23: Simulated boulder positions in red, and rover path in blue.

Figure 4.24: Rover localisation estimate using cylinder model mask matching.
Figure 4.25: Rover localisation estimate using dome model mask matching.

Figure 4.26: Absolute error over traverse using dome and cylinder model mask matching.
such as PM-SLAM, the benefits could increase, and the issues with assigning features to the wrong global map boulders at longer ranges could be addressed.

## 4.7 Conclusions and Future Work

In this chapter, a system has been developed and results from a limited test analysed. The algorithm was designed to associate image descriptors from monocular images with features detected in a global map. The global map was presumed to have been generated offline in advance of any mission and pre-loaded into the rover’s memory. The orbiter mask matching system was shown to be able to provide a localisation estimate, offering accuracy comparable to other monocular VO systems.

The algorithm was tested using two options for the 3-dimensional model used to represent the orbital features as boulders in the rover images. The first model used cylinders to represent the boulders (Section 4.4.1) based on existing boulder modelling techniques, and the second represented them as domes (Section 4.4.2), which was a novel evolution introduced in this work. Using these models, the point-based features detected in the monocular images taken in the rover were projected into 3-dimensions using a modified version of the direct depth algorithm (Section 4.3), giving the features co-ordinates in the rover’s local co-ordinate frame (Sections 4.5.1 and 4.5.2).

An example 190 metre traverse was used to evaluate the use of the orbiter mask matching algorithm for localising a rover on a planetary surface. The test dataset consisted of colour images generated from a PANGU simulation in a 250 metre by 200 metre boulder field. The localisation was performed by using the features generated by the orbiter mask matching algorithm in a framework similar to VO systems, but with the added benefit of simultaneously adding feature descriptors to the features in the global map, making them trackable.

Using a cylinder model to represent the boulders was found, in this test, to provide a higher accuracy of localisation estimate compared with using the dome model, however the estimates from both did drift over the length of the traverse, and therefore were not able to provide an accurate absolute localisation.
In order to achieve a more accurate absolute localisation, the algorithm developed in this section could be used to add functionality to a planetary SLAM system, like the PM-SLAM system described in Chapter 3. This could reduce the overall localisation drift, compared to a VO system and also build a trackable global map. The trackable features in the global map could also be used to help teams of rovers working together to explore the planetary features. One rover encountering a unknown feature, which has been previously encountered by another rover, could identify it and therefore determine it’s position in the global map.

Improvements in projection of the feature points from the image plane to a 3-dimensional representation of the boulders could be improved using shape from shading methods in order to determine the relative heights of the landscape visible to the rover. This would allow more accurate positioning of the features in 3-dimensional space.
Summary and Further Work

5.1 Thesis summary

This thesis investigates autonomous localisation techniques for planetary exploration rover missions. The research work carried out towards this goal was based on the premise of allowing rovers to complete long traverses in sample return type mission scenarios, with minimum intervention from terrestrial based ground stations. Novel methods were developed towards this goal, using existing SLAM techniques and newer methods incorporating orbital image data and were tested on various representative datasets.

Chapter 1 served as an introduction to the thesis, comprising the motivation of this research work (i.e. the research benefits), the problem statement, the complexity of the required system and an introduction to the research contributions. The main objective of the thesis was to present state-of-the-art analysis and techniques for performing autonomous planetary localisation using monocular imagery, taken by an exploration rover on the Martian surface. The chapter concluded with a breakdown of the principal contributions.
In Chapter 2, a detailed literature survey was presented. The survey detailed the current state of the art for planetary localisation of rovers, and the capabilities of the systems flown on current and past Mars exploration missions. A detailed description of SLAM techniques, and the components of a SLAM system were explained. Two different classes of image feature detection were described, and theory behind example algorithms of each type were explained. The final section of the chapter presented methods for using orbital imagery to aid in planetary localisation. Chapter 3 and Chapter 4 present the principle methodological contributions, comprising the proposed solutions to the research problem.

The development of a SLAM system for use in planetary exploration is described in Chapter 3. The challenges of using traditional point-based image features on board systems with low computational capabilities was demonstrated, with the trade off between accuracy and processing time per step demonstrated over a simulated dataset. A hybrid-salient feature detection algorithm was used, in order to cluster many feature-based points in a scene into a few salient-blobs, whilst maintaining trackability. The clustering also functioned as an outlier rejection technique. The hybrid-salient feature tracking system was tested using the MOTA measure of accuracy. The algorithm was shown to provide robust tracking over a set of real-world images captured in the Atacama desert. The algorithm was also tested under varying illuminations in order to further demonstrate its robustness. A geometric method for retrieving feature depth from monocular images, Direct Depth, was developed by extending an existing method to give co-ordinates in the real-world x-y plane.

Direct depth and hybrid-salient feature detection were combined with a modular SLAM system to create PM-SLAM, a monocular SLAM for planetary exploration. PM-SLAM was evaluated against three datasets, one simulated and two captured by rovers in planetary analogous environments. PM-SLAM was tested using several combinations of point-based feature, salient feature and SLAM filter against a number of metrics. PM-SLAM was able to successfully loop-close, and provide localisation with better accuracy than a state-of-the-art VO system designed for planetary localisation. A combination of an EKF, with SURF-Hou hybrid saliency was found to provide the best combination of accurate localisation and quick computation.
Chapter 4 introduced a proof of concept of the Orbiter Mask Matching algorithm. Feature descriptors detected from rover images were associated with features extracted from orbiter images into a global map. In order to represent the 2-dimensional image points in 3-dimensional space, two models were developed in parallel. The first represented the boulders as cylinders and the second represented the boulders as domes. The Direct Depth algorithm developed in Chapter 3 is modified to allow projection of image points onto the boulder models, allowing them to be associated with orbiter map features and tracked.

The Orbiter Mask Matching algorithm was evaluated over a 190 metre traverse through a simulated boulder field using a VO type framework to calculate the movement between successive frames. The resulting localisation provided comparable accuracy to other monocular VO systems, while also allowing orbiter features to be associated with descriptors from the rover images. However the position estimates produced did exhibit drift over the length of the traverse, and therefore did not provide an accurate absolute localisation.

5.2 Future work

The datasets available for testing planetary localisation were limited, especially for long traverses with robust ground truths. Datasets comprising orbital or aerial imagery were also limited, and hence simulations were used for developing Orbiter Mask Matching. Evaluation against larger datasets would enable the robustness of these algorithms to be further explored.

The modular nature of the hybrid-salient features used in Chapter 3 allow the continual update of the system, as new saliency and point-based feature algorithms are developed. This would allow hybrid-saliency to benefit from any improvements in feature detection accuracy and descriptor robustness. Given the sparse and homogeneous nature of the Martian surface in the visible spectrum, passive infra-red images could also be used, with saliency applied to these images in order to better separate objects of interest from the background. This data could also be used for characterising the geological
significance of features in the rover visual field creating an even richer map for planetary exploration.

In Chapter 4, the Orbital Mask Matching technique was demonstrated as part of a VO type framework in order to localise the rover over each traverse step. By incorporating Orbiter Mask Matching into the PM-SLAM system from Chapter 3, the localisation drift could be reduced by the probabilistic SLAM filter, whilst also retaining the benefits of providing descriptors in the global map. Such a global map could also be used for co-localising teams of rovers in a multi-rover mission scenario. By attaching descriptors to objects in the global map, a rover would be able to identify features encountered by another rover previously.
Appendices
Experimental Datasets

Subsets from four datasets have been used in order to perform experimental evaluation in this thesis. Two datasets were simulated using the PANGU tool and the remaining two were captured in sandy rocky terrains by test rovers.

A.1 PANGU D-shaped localisation dataset

This dataset was created using the PANGU tool with boulder and terrain distribution defined by presets in the PANGU GUI. The traverse was designed to be shaped like D in order to include straight line portions as well as gentle and severe turns while also allowing the rover to loop close.

1. **Total count:** 479 control steps and corresponding image frames over 60.1 metres.

2. **Control signal:** 479 position deltas comprising the distance moved and the change in heading angle.
3. **Ground truth**: Control signal with added zero-mean Gaussian noise. A standard deviation of 0.05m for the distance travel and 0.005rad for the heading change were used.

4. **Image capture**: The control signal was used to position the “camera” at each step and a greyscale image was saved.

5. **Image resolution**: 512x512 pixels.

6. **Camera height**: 2 metres.

7. **Camera field of view**: $65^\circ \times 65^\circ$.

8. **Camera pitch**: $30^\circ$.

### A.2 West Wittering localisation dataset

This dataset was recorded using the SMART platform in West Wittering, on the south coast of England by the STAR Lab. The rover was manually teleoperated around a sandy terrain with an array of synthetic and real boulders and recorded colour images.

1. **Total count**: 172 control steps and corresponding image frames over 10.8 metres.

2. **Control signal**: 172 position deltas comprising the distance moved and the change in heading angle, computed from the teleoperation logs. The control signals and captured images were synched using the ROS timestamping functionality.

3. **Ground truth**: A GPS unit on board SMART was used to capture the localisation ground-truth. The longitude and latitude were recorded by ROS and therefore carried the same timestamp data as the images and control signals.

4. **Image capture**: Video was captured at 24 frames per second by a standard webcam mounted on the rover. These images were filtered by timestamp to coincide with the control signals.
A.3 SEEKER localisation dataset

This dataset was recorded as part of an ESA rover trial in the Atacama desert. Grayscale images were captured as the rover traversed. For the purposes of the evaluation in this thesis a subset of the data was selected that was representative of a planetary dataset. The incomplete nature of the available GPS data for groundtruthing required a sampled and averaged groundtruth to used.

102 metre traverse was chosen, which contained 2711 images which were 512 x 384

1. **Total count**: 2711 control steps and corresponding image frames over 102 metres.

2. **Control signal**: 2711 position deltas comprising the distance moved in the x and y directions. The control data was timestamped to allow synching with the captured images.

3. **Ground truth**: A GPS unit used to capture the localisation ground-truth. This data was also timestamped, but external factors including the presence of a generator at the test-site have affected the readings at some points. Sudden unfeasible jumps, which do not correlate with the images, have been smoothed and interpolated.

4. **Image capture**: Video was captured camera mounted on the rover. These images were filtered by timestamp to coincide with the control signals.

5. **Image resolution**: 512x384 pixels.

6. **Camera height**: 1.7 metres.
7. Camera field of view: 53.14° x 45°.


A.4 SEEKER MOTA dataset

A second subset of the SEEKER dataset was used for MOTA analysis. The dataset was captured using the same camera and setup as that in Appendix A.3. The images were manually annotated, with a bounding box drawn around large boulders and an identifying index used to track the same feature across successive images.

1. **Total count:** 1136 images with annotated boulders.

2. **Annotated feature count:** 36.

3. **Total detected feature count:** 4123.

4. **Image resolution:** 512x384 pixels.

5. **Camera height:** 1.7 metres.

6. **Camera field of view:** 53.14° x 45°.

7. **Camera pitch:** 29.4°.

A.5 PANGU straight line traverse

This dataset was created using the PANGU tool with boulder and terrain distribution defined from a configuration provided by Airbus Defence and Space for ExoMars testing. The frequency of large diameter boulders was increased to test orbital matching. The traverse was a straight line through a large boulder field, with colour images captured at each step.

1. **Total count:** 190 control steps and corresponding image frames.
2. **Control signal:** 190 position deltas comprising the distance moved and the change in heading angle.

3. **Ground truth:** Control signal with added zero-mean Gaussian noise. A standard deviation of 0.05m for the distance travel and 0.005rad for the heading change were used.

4. **Image capture:** The control signal was used to position the “camera” at each step and an RGB image was saved.

5. **Image resolution:** 512x512 pixels.

6. **Camera height:** 2 metres.

7. **Camera field of view:** 65° x 65°.

8. **Camera pitch:** 30°.
APPENDIX B

PM-SLAM Software Architecture

B.1 Package Structure

PM-SLAM is composed of 9 main packages. Where possible the code has been designed
to be fully modular and separable to allow additional filters and algorithms to be
added to each package with minimal requirement for integrating them. In order to
facilitate this, an abstract factory pattern was used throughout together with
ROS messaging and service infrastructure to standardise expected inputs, outputs and
communication between modules. The packages, and the various modules that they
contain are described in this appendix.
B.2 PMSLAM-filters

\begin{table}[h]
\centering
\begin{tabular}{|c|l|}
\hline
\textbf{pmslam\_filters} & \textbf{SLAMNode} \\
\hline
ROS & SLAMNode \\
SLAM & SLAMFilterFactory \\
SLAM & SLAMFilter \\
SLAM & \text{EKFFilter} \\
SLAM & \text{EIFFilter} \\
SLAM & \text{fastSLAM} \\
MAP & map \\
MAP & mapFeature \\
MAP & features3D \\
\hline
\end{tabular}
\caption{Figure B.1: Modules in the pmslam\_filters package. These modules all contribute to the implementation of the SLAM filters. Modules marked with ROS are those that provide an interface with the ROS architecture of PM-SLAM as a whole. Modules marked with SLAM are those that implement the filters and their wrappers. Modules marked with MAP are classes for storing map and feature data.}
\end{table}

**B.2.1 SLAMNode**

The SLAMNode module acts as a ROS wrapper, for the SLAM filters. It is used in conjunction with ROS launch files to specify which filter to use and the starting parameters such as initial rover pose. Once the SLAM filter has been initialised the SLAMNode starts a service implementing the SLAMFilterService 2.4.5 which takes control and feature data and returns an updated rover pose.

**B.2.2 SLAMFilterFactory**

This modules abstracts creation of a SLAM Filter. A constant can be used to specify the type of filter needed. New filters can be easily implemented as long as they are subclassed from SLAM Filter (Section B.2.3).
B.2.3 SLAMFilter

The base class for all SLAM filters used in PM-SLAM. This class implements the vehicle model from Section 2.2.2 and the measurement model from Section 2.2.3, as well as abstract definitions for the predict, update and augment steps, which are intended to be overridden by implemented subclasses.

B.2.4 EKFFilter

Implementation of the EKF algorithm adapted from MATLAB code by Tim Bailey and Juan Nieto. Subclassed from SLAMFilter (B.2.3).

B.2.5 EIFFilter

Implementation of the EIF algorithm adapted. Subclassed from SLAMFilter (B.2.3).

B.2.6 FastSLAM

Implementation of the FastSLAM 2 algorithm adapted from MATLAB code by Tim Bailey and Juan Nieto. Subclassed from SLAMFilter (B.2.3).

B.2.7 Map

Structure for storing map of features observed by rover and added during the augment step. Each feature is stored as a mapFeature (Section B.2.8).

B.2.8 MapFeature

Structure for storing each feature observed by rover and added during the augment step.
B.2.9 Features3D

Feature position structure used for features in the rover’s local co-ordinate frame.

B.3 PMSLAM-HybridFeatureDetection

<table>
<thead>
<tr>
<th>pmslam_hybridfeaturedetection</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROS hybridFeatureMatcher</td>
</tr>
<tr>
<td>ROS blobExtractor</td>
</tr>
<tr>
<td>ROS featureMatcher</td>
</tr>
<tr>
<td>ROS depthExtractor</td>
</tr>
<tr>
<td>Saliency saliencyMapFactory</td>
</tr>
<tr>
<td>Saliency saliencyMap</td>
</tr>
<tr>
<td>Saliency saliencyMapHou</td>
</tr>
<tr>
<td>Saliency saliencyMapRudinac</td>
</tr>
<tr>
<td>Point featureExtractorFactory</td>
</tr>
<tr>
<td>Point featureExtractor</td>
</tr>
<tr>
<td>Point featureExtractorSURF</td>
</tr>
<tr>
<td>Point featureExtractorSIFT</td>
</tr>
<tr>
<td>Point featureExtractorBRISK</td>
</tr>
<tr>
<td>Depth directDepth</td>
</tr>
</tbody>
</table>

**Figure B.2:** Modules in the pmslam_hybridfeaturedetection package. These modules all contribute to the implementation of Hybrid Feature Detection. Modules marked with ROS are those that provide an interface with the ROS architecture of PM-SLAM as a whole. Modules marked with Saliency are those that implement the saliency algorithms and their wrappers. Modules marked with Point are those that implement the point-based feature detection algorithms and their wrappers. Modules marked with Depth implement depth detection algorithms.
B.3.1 HybridFeatureMatcher

ROS wrapper for all the services used by hybrid feature matching. It is used in conjunction with ROS launch files to specify which saliency, point-based feature and depth algorithms to use as well as initial parameters. Once each of the modules has been initialised the hybridFeatureMatcher starts the HybridFeatureMatch (Section 2.4.4) service. As image data is provided by the router module (Section 2.6.1) the HybridFeatureMatcher requests the ExtractBoundingBoxes service (Section 2.4.1), ExtractFeatureMatches service (Section 2.4.3) and ExtractFeatureDepths service (Section 2.4.2) in order.

2.3.2 BlobExtractor

The blobExtractor is a ROS wrapper for the salient feature extraction part of hybrid feature matching. It listens for requests on the ExtractBoundingBoxes service, and then using the selected saliency algorithm provides a list of bounding boxes for salient features in the image plane.

2.3.3 FeatureMatcher

The featureMatcher is a ROS wrapper for the modules that extract and match point based features within the salient bounding boxes. It returns the unique id of each feature and a flag denoting if they are new features.

2.3.4 DepthExtractor

The depthExtractor is a ROS wrapper for depth extraction modules. It returns the computed local depth of salient bounding boxes.

2.3.5 SaliencyMapFactory

This modules abstracts creation of a salient feature extractor. A constant can be used to specify the type of algorithm used. New algorithms can be easily implemented as
long as they are subclassed from SaliencyMap (Section 2.3.6).

2.3.6 SaliencyMap

The base class for all salient feature extraction algorithms used by hybrid feature detection. This class abstracts the definition for the calculateSalientMap function, which is intended to be overridden by implemented subclasses.

2.3.7 SaliencyMapHou

This is an implementation of the Hou saliency algorithm from [45].

2.3.8 SaliencyMapRudinac

This is an implementation of the Rudinac saliency algorithm from [86].

2.3.9 FeatureExtractorFactory

This module abstracts creation of a point-based feature extractor. A constant can be used to specify the type of algorithm used. New algorithms can be easily implemented as long as they are subclassed from featureExtractor (Section 2.3.10).

2.3.10 FeatureExtractor

The base class for all point-based feature extraction algorithms used by hybrid feature detection. This class implements a function for adding new point-based features to the database of features as well as the matching algorithm in Section 3.3.4.

2.3.11 FeatureExtractorSURF

This module uses the OpenCV2 [25] implementation of the SURF algorithm [12].
2.3.12 FeatureExtractorSIFT

This module uses the OpenCV2 [25] implementation of the SIFT algorithm [50].

2.3.13 FeatureExtractorBRISK

This module uses the OpenCV2 [25] implementation of the BRISK algorithm [56].

2.3.14 DirectDepth

This module implements the Direct Depth algorithm described in Section 3.4.

2.4 PMSLAM-Services

<table>
<thead>
<tr>
<th>pmslam_services</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROS ExtractBoundingBoxes</td>
</tr>
<tr>
<td>ROS ExtractFeatureDepths</td>
</tr>
<tr>
<td>ROS ExtractFeatureMatches</td>
</tr>
<tr>
<td>ROS HybridFeatureMatch</td>
</tr>
<tr>
<td>ROS SLAMFilterService</td>
</tr>
</tbody>
</table>

Figure 2.3: Services used by PM-SLAM

2.4.1 ExtractBoundingBox

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensor_msgs/Image inputImage</td>
<td>pmslam_messages/BoundingBox[] boundingBoxes</td>
</tr>
</tbody>
</table>

Table 2.1: ExtractBoundingBox service parameters
2.4.2 ExtractFeatureDepths

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pmslam_messages/BoundingBox[]</code></td>
<td><code>geometry_msgs/Point32[]</code> featureDepths</td>
</tr>
</tbody>
</table>

Table 2.2: ExtractFeatureDepths service parameters

2.4.3 ExtractFeatureMatches

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sensor_msgs/Image</code> inputImage</td>
<td><code>pmslam_messages/ImageFeature[]</code> imageFeatures</td>
</tr>
<tr>
<td><code>pmslam_messages/BoundingBox[]</code></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.3: ExtractFeatureMatches service parameters

2.4.4 HybridFeatureMatch

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sensor_msgs/Image</code> inputImage</td>
<td><code>pmslam_messages/Feature3D[]</code> features</td>
</tr>
</tbody>
</table>

Table 2.4: HybridFeatureMatch service parameters
2.5. PMSLAM-Messages

2.4.5 SLAMFilterService

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>geometry_msgs/Point32 controlVector</td>
<td>geometry_msgs/Pose2D roverPose</td>
</tr>
<tr>
<td>pmslam_messages/Feature3D[] features</td>
<td>std_msgs/Time predictStart</td>
</tr>
<tr>
<td>std_msgs/Time predictEnd</td>
<td></td>
</tr>
<tr>
<td>std_msgs/Time updateStart</td>
<td></td>
</tr>
<tr>
<td>std_msgs/Time updateEnd</td>
<td></td>
</tr>
<tr>
<td>std_msgs/Time augmentStart</td>
<td></td>
</tr>
<tr>
<td>std_msgs/Time augmentEnd</td>
<td></td>
</tr>
<tr>
<td>float64 roverError</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.5: SlamFilterService service parameters

2.5 PMSLAM-Messages

Figure 2.4: Messages used by PM-SLAM

2.5.1 BoundingBox

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>int32</td>
<td>index</td>
</tr>
<tr>
<td>geometry_msgs/Point32</td>
<td>topleft</td>
</tr>
<tr>
<td>geometry_msgs/Point32</td>
<td>bottomright</td>
</tr>
</tbody>
</table>

Table 2.6: BoundingBox message definition
2.5.2 Feature3D

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>int32</td>
<td>featureID</td>
</tr>
<tr>
<td>geometry_msgs/Point32</td>
<td>featurePosition</td>
</tr>
<tr>
<td>pmslam_messages/BoundingBox</td>
<td>boundingBox</td>
</tr>
<tr>
<td>bool</td>
<td>exists</td>
</tr>
</tbody>
</table>

Table 2.7: Feature3D message definition

2.5.3 ImageFeature

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>pmslam_messages/BoundingBox</td>
<td>boundingBox</td>
</tr>
<tr>
<td>int32</td>
<td>featureID</td>
</tr>
<tr>
<td>bool</td>
<td>exists</td>
</tr>
</tbody>
</table>

Table 2.8: ImageFeature message definition

2.6 PMSLAM-router

Figure 2.5: Modules in the pmslam_router package. This module implements the synchronisation of sensor data using ROS.

2.6.1 Router

This module implements the synchronisation of sensor data using built in ROS synchronisation strategies based on the data timestamps. This groups odometry and image data together for each SLAM step.
2.7 Other PM-SLAM Modules

2.7.1 PM-SLAM data

The PM-SLAM data module was used for offline tests. Control data and odometry data from CSV files are combined with image files to create inputs for the PM-SLAM system.

2.7.2 PM-SLAM GUI

The PM-SLAM GUI module uses the Qt based GUI tools included with ROS. This was used for live monitoring of simulations.

2.7.3 PM-SLAM launch

The PM-SLAM launch module was used for to provide launch parameters and configurations for each run of PM-SLAM. This included choice of SLAM filter algorithm, saliency algorithm and point-based feature algorithm.

2.7.4 PM-SLAM results

The PM-SLAM results module was used for recording the output of PM-SLAM for offline analysis. This included the rover pose data along with timing data for each of the processes.


[95] Nuno Silva, Richard Lancaster, and Jim Clemmet. ExoMars rover vehicle mobility functional architecture and key design drivers. In *Proceedings of the 12th Sym-


