Edge Feature and Optical Flow Terrain Aid for GNSS-Denied Airborne Visual Navigation

By
David Garner Williams

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

Department of Aerospace, Mechanical and Mechatronic Engineering
Faculty of Engineering and Information Technologies
The University of Sydney

December 2016
Declaration

This is to certify that to the best of my knowledge, the content of this thesis submission is my own work. This submission contains no material written by another author, nor which has been submitted for any degree or other purposes, unless otherwise indicated.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

This thesis contains material that is to be published in [1–3], of which I am the author of, as well as the sole writer, and of which I developed all presented contributions and results. The co-author of these works, Peter W. Gibbens, provided supervision, guidance, and assisted with the editing of these works.

David G. Williams

17th December, 2016
Acknowledgements

I would like to thank my supervisor, Associate Professor Peter Gibbens, for his guidance and support throughout my postgraduate journey. The different perspectives he has provided, as well as his constant drive for improvement, has made a significant impact on my thesis. The time and effort he has dedicated to our weekly meetings, as well as reading through my thesis and journal papers has been invaluable. I am also very grateful to Steven Dumble, who developed the aerial test platform which collected the real flight data used to verify the work presented in this thesis. Furthermore, his assistance and guidance in the initial stages of my thesis were greatly appreciated.

I would like to thank my fellow Aero postgraduate students at USyd for your friendship and support throughout my studies. Finally, I would like to dedicate this thesis to my family, especially my parents. The unwavering support and encouragement you have shown me has been indispensable.
Abstract

Conventional autonomous aircraft navigation relies on an inertial measurement unit and Global Navigation Satellite System (GNSS) to correct for time-accrued dead-reckoned drift. Inertial measurements are used to accurately track fast changes, whereas GNSS signals prevent the localisation solution from deteriorating over time due to inertial drift. This reliance on GNSS risks disorientation of the autonomous vehicle due to GNSS failure from atmospheric events, jamming or satellite destruction. This would result in an inability to navigate through GNSS-denied environments. Navigation methods which do not require GNSS predominantly rely on active sensors, such as radar or laser range finders, which impose critical platform restrictions due to weight, volume and power draw. Furthermore, reliance on active sensors will prevent any stealthy operations of an aerial vehicle.

For a human pilot, vision plays a major role in the methods by which they use to navigate an aerial vehicle. These navigation techniques, such as Visual Flight Rules enable a pilot to localise an aircraft without reliance on GNSS or any active sensors by recognising visually distinct features. These features, or landmarks, may be associated with a feature map to determine location, orientation, altitude and speed. Visually distinct features may include point landmarks such as buildings, however will more generally consist of curve features, such as lake or river edges, forest boundaries or roads. Furthermore, a pilot may also use the shape of underlying terrain features such as hills and valleys as localisation aids.

This thesis outlines the development and implementation of a computer vision based autonomous navigation system. This system employs the same visual navigation techniques a human pilot would, using curve and surface based landmarks. Visual information is fused with inertial information in a probabilistic state filter in order to minimise the dead reckoning drift from pure inertial integration. This visual system is separated into two main localisation research sections; curve feature tracking and terrain surface matching. Curve feature tracking allows curves such as edges to be estimated, tracked and associated with known features for position updates. Terrain surface matching involves the estimation of the local terrain profile. This estimate may then be associated with a known terrain profile, such as a digital terrain elevation map (DTEM), for position updates.

The Simultaneous Localisation and Mapping (SLAM) process is a method by which newly detected features may be used for navigation, even through unknown environments. The first research area of this thesis involves the application of the SLAM algorithms to spline features in the full six degrees of freedom aerial navigation case. Image processing techniques are outlined which allow regions of differing ground type to be determined, in order to detect edge curve features. The methods by which splines are used to characterise features, and the mechanics of how they are updated through multiple measurements, are outlined in detail. Furthermore, this thesis shows how the range to detected features may be estimated. Methods which help to improve the robustness of the feature detection algorithms, as well as the estimated SLAM spline feature map and data associations, are also outlined.
Although navigation in unknown environments is an interesting topic of research, there are few real-world environments which are truly completely unknown. A number of freely available databases, such as Google Earth, provide convenient sources of information which may be used to localise an aircraft. This thesis demonstrates the segmentation of aerial imagery retrieved from Google Earth into edge features. These features are then assembled into a data-compact database of spline features, which is used for Visual Terrain Aided Navigation (VTAN), allowing accurate localisation of the aerial vehicle.

The second research area of this thesis involves the use of Digital Terrain Elevation Maps (DTEM). Some of these maps describe the shape of most of the Earth’s landmass, such as that produced by the Shuttle Radar Topography Mission. These describe the terrain profile, such as hills, cliffs and valleys. For human pilots, this information is often provided in the form of a topography map. A pilot can estimate the shape of surrounding terrain by observing changes in the apparent relative speed of terrain as it moves by. Comparisons may then be made between this and the topography map, helping to localise the vehicle. This thesis presents a method by which movement rates of imagery in a camera frame (optical flow) can be used to estimate the shape of underlying terrain. The estimated terrain contour shape is then compared to a freely available DTEM, allowing localisation of the vehicle. Optical flow is simultaneously used to provide visual odometry information, estimating the vehicle velocity.

Some visually distinct features may change over time, such as water body boundaries changing with varying water height. This can cause navigation problems, as localising using these features may give biased position estimates. Considering river or lake edge changes can be expected to follow the shape of the terrain, a Digital Terrain Map may be used to predict this change. Should changes in water level be known (such as through water storage reports) changes in edge feature location can be accounted for, as demonstrated in this thesis. This improves the reliability of the presented visual navigation system. Conversely, should the water level change not be known, a method of using the terrain map to estimate this offset is established. This allows the possibility of aerial monitoring of remote river or lake ecosystems.

The outlined system is demonstrated to greatly improve navigation accuracy in unknown environments using real flight test data. Terrain assistance associations with a known curve feature map are shown to provide visual localisation accuracies of under 20 metres. Finally, optical flow based terrain contour matching techniques result in localisation accuracies approaching those of edge feature terrain assistance, without requiring the use of a pre-processed feature map.
Contents

1 Introduction .................................................. 1
   1.1 Background ............................................... 1
   1.2 Problem Statement ........................................ 4
   1.3 Contributions ............................................ 7
   1.4 Test Platform ............................................. 8
   1.5 Chapter Outline .......................................... 11

2 Related Work ................................................ 13
   2.1 Introduction .............................................. 13
   2.2 Commercial Aerial Navigation Aids ....................... 15
      2.2.1 Ground Based Radio Systems ....................... 15
      2.2.2 Space Based Radio Systems ....................... 16
      2.2.3 Observations ....................................... 17
   2.3 Sensors & Data Fusion ................................... 17
      2.3.1 Commonly Available Sensors ....................... 17
      2.3.2 Data Fusion Systems ............................... 19
      2.3.3 Linear Kalman Filter .............................. 20
      2.3.4 Extended Kalman Filter (EKF) .................... 20
      2.3.5 Unscented Kalman Filter (UKF) ................. 21
      2.3.6 Particle Filter .................................... 21
      2.3.7 Information Filter ................................ 22
2.3.8 Simultaneous Localisation and Mapping (SLAM) .................................. 22
2.3.9 Terrain Assisted SLAM (TAN-SLAM) ............................................. 23
2.3.10 Observations ..................................................................................... 23

2.4 Contour Based Aerial Navigation Methods ........................................... 24
2.4.1 Contour Based Terrain Aided Navigation (TAN) ................................. 24
2.4.2 Visual Contour Based TAN ................................................................. 25
2.4.3 Horizon Based TAN ............................................................................ 26
2.4.4 Observations ..................................................................................... 27

2.5 Visual Feature Detection ......................................................................... 28
2.5.1 SIFT/SURF Methods ......................................................................... 28
2.5.2 Texture Segmentation ........................................................................ 29
2.5.3 Deep Learning Methods ..................................................................... 32
2.5.4 Observations ..................................................................................... 33

2.6 Feature Based Aerial Navigation Techniques ......................................... 35
2.6.1 Point Feature Methods ...................................................................... 35
2.6.2 Edge Feature Methods ...................................................................... 37
2.6.3 Map Aided Navigation ....................................................................... 37
2.6.4 Terrain Aided SLAM Fusion ............................................................... 39
2.6.5 Observations ..................................................................................... 40

2.7 Data Association .................................................................................... 41
2.7.1 Spline Node Matching ...................................................................... 41
2.7.2 Spline Curvature Matching ................................................................. 42
2.7.3 Observations ..................................................................................... 44

2.8 Summary ............................................................................................. 44

3 Background Theory .................................................................................. 48
3.1 Reference Systems and Coordinate Transformations ............................. 48
3.1.1 Aerial Reference Systems ................................................................. 49
3.1.2 Visual Reference Systems ........................................... 50
3.1.3 Axis Rotation Transforms ........................................ 51

3.2 Image Processing ......................................................... 52
3.2.1 Texture Segmentation .............................................. 53
3.2.2 Watershedding Algorithm ......................................... 53
3.2.3 Maximum Likelihood Classification ................................. 53
3.2.4 Training Data .......................................................... 54
3.2.5 Over-segmentation .................................................. 54
3.2.6 Feature Extraction ................................................... 55

3.3 Spline Curve Modelling .................................................. 56
3.3.1 Introduction to B–Splines ........................................... 57
3.3.2 Mathematical Spline Model ......................................... 57
3.3.3 Spline Fitting .......................................................... 58
3.3.4 Closest Point On Spline ............................................. 59
3.3.5 The Non-Uniform Rational B-Spline (NURBS) .................... 61
3.3.6 Spline Curve Association ........................................... 61

3.4 Parameter Estimation ..................................................... 63
3.4.1 Solution Covariance .................................................. 64
3.4.2 Ordinary Least Squares ............................................ 64
3.4.3 Weighted Least Squares ........................................... 65
3.4.4 Generalised Least Squares ........................................ 65
3.4.5 Chi-Square Test ........................................................ 65

3.5 Extended Kalman Filter .................................................. 66
3.5.1 Prediction Step ......................................................... 67
3.5.2 Update Step ............................................................. 68

3.6 Simultaneous Localisation and Mapping ............................... 68
3.6.1 Feature State Initialisation .......................................... 69
3.6.2 Kalman Prediction Modification ........................................ 70
3.6.3 Kalman Update Modification ........................................ 71
3.7 State and Sensor Models .................................................. 72
  3.7.1 Vehicle Process Model ................................................. 72
  3.7.2 Observation Models .................................................... 73
3.8 Optical Flow ............................................................... 80
  3.8.1 Methodology ............................................................. 80
  3.8.2 Decoupling Relative Motion ......................................... 80
3.9 System Overview and Summary .......................................... 82
  3.9.1 Inertial Navigation ...................................................... 82
  3.9.2 Edge Feature Detection ............................................... 82
  3.9.3 Visual Spline SLAM .................................................... 83
  3.9.4 Terrain Aided Navigation .............................................. 83
  3.9.5 Optical Flow Based Odometry ........................................ 84
  3.9.6 Visual Terrain Profile Matching ...................................... 84

4 Image Processing & Feature Detection .................................. 85
  4.1 Ground Appearance ..................................................... 87
  4.2 Image Segmentation ..................................................... 88
  4.3 Classifying Textures ..................................................... 89
    4.3.1 Texture Parametrisation .......................................... 89
    4.3.2 Parameter Variance ................................................ 92
    4.3.3 Training Data ........................................................ 93
  4.4 Reducing Over-segmentation ........................................... 94
    4.4.1 Gaussian Filtering: .................................................. 95
    4.4.2 Gradient Threshold: ................................................ 97
    4.4.3 Adjacent Region Similarity: ....................................... 98
    4.4.4 Texture Type Merge: ............................................... 100
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5</td>
<td>Edge Feature Extraction</td>
<td>100</td>
</tr>
<tr>
<td>4.6</td>
<td>Sun Reflection Mask</td>
<td>102</td>
</tr>
<tr>
<td>4.7</td>
<td>Results</td>
<td>104</td>
</tr>
<tr>
<td>4.8</td>
<td>Summary</td>
<td>108</td>
</tr>
<tr>
<td>5</td>
<td>Spline-Based Visual SLAM</td>
<td>110</td>
</tr>
<tr>
<td>5.1</td>
<td>Proposed Spline SLAM System</td>
<td>111</td>
</tr>
<tr>
<td>5.2</td>
<td>Range Prediction from Terrain</td>
<td>114</td>
</tr>
<tr>
<td>5.3</td>
<td>Spline End Clamping</td>
<td>118</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Spline End Re-Weighting</td>
<td>119</td>
</tr>
<tr>
<td>5.4</td>
<td>Kalman Innovation Methodology</td>
<td>120</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Point-Spline vs Spline-Spline Association</td>
<td>120</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Point-Spline vs Spline-Spline Updates</td>
<td>122</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Data Association</td>
<td>124</td>
</tr>
<tr>
<td>5.4.4</td>
<td>Kalman Innovation</td>
<td>125</td>
</tr>
<tr>
<td>5.4.5</td>
<td>Map Optimisation</td>
<td>126</td>
</tr>
<tr>
<td>5.5</td>
<td>Results</td>
<td>126</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Statistical Analysis</td>
<td>126</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Navigation Solution Accuracy</td>
<td>127</td>
</tr>
<tr>
<td>5.5.3</td>
<td>SLAM Computation Times</td>
<td>131</td>
</tr>
<tr>
<td>5.6</td>
<td>Summary</td>
<td>133</td>
</tr>
<tr>
<td>6</td>
<td>Terrain Feature Assisted Navigation</td>
<td>135</td>
</tr>
<tr>
<td>6.1</td>
<td>Feature Database Creation</td>
<td>137</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Google Earth Imagery</td>
<td>137</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Texture Segmentation</td>
<td>138</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Imagery Inconsistency</td>
<td>138</td>
</tr>
<tr>
<td>6.2</td>
<td>Edge Data Association</td>
<td>140</td>
</tr>
</tbody>
</table>
6.2.1 Gaussian Filtering ........................................ 140
6.2.2 Adaptive Filtering ........................................ 140
6.2.3 Shape Wavelength ........................................ 143
6.2.4 Tokenisation .............................................. 143
6.2.5 Over-Representation ..................................... 144
6.2.6 Candidate Recombination ................................ 145
6.2.7 Critical Points ........................................... 146
6.2.8 Critical Region Comparison .............................. 146
6.2.9 Point Match Regression .................................. 147
6.2.10 Choose Best Match ...................................... 148
6.3 Kalman Filter Sensor Fusion ................................. 148
  6.3.1 Map Match Fusion ...................................... 148
  6.3.2 Search Space Prediction ................................ 149
6.4 Feature Estimation Area .................................... 151
6.5 Compensating for Water Level Changes .................. 153
6.6 Water Level Estimation .................................... 154
6.7 Results ..................................................... 155
  6.7.1 Spline Feature Database ................................. 156
  6.7.2 Statistical Analysis ..................................... 157
  6.7.3 State Estimation Area and Sensor Variance .......... 157
  6.7.4 TAN-SLAM Accuracy ................................ 160
  6.7.5 Computation Time ..................................... 163
  6.7.6 TAN Without SLAM Comparison .................... 166
  6.7.7 Water Level Adjustment and Estimation ............ 167
6.8 Summary .................................................... 167

7 Terrain Profile Assisted Navigation ......................... 169
  7.1 Optical Flow Overview ................................... 170
7.2 Optical Flow Calculation .................................................. 172
  7.2.1 Optical Flow Prediction .............................................. 173
  7.2.2 Flow Update .......................................................... 173
7.3 Optical Contour Association ............................................. 175
  7.3.1 Range from Flow ...................................................... 176
  7.3.2 Terrain Contour Estimate ........................................... 177
  7.3.3 Contour Association .................................................. 181
  7.3.4 Vehicle Position Update ............................................ 182
7.4 Visual Odometry ............................................................. 183
  7.4.1 Velocity Update ....................................................... 183
7.5 DTEM Water Edge .......................................................... 184
  7.5.1 Water Edge Estimation .............................................. 185
  7.5.2 Terrain Water Edge Innovation .................................... 185
7.6 Results ................................................................. 187
  7.6.1 Verification of Optical Flow Aids ................................ 187
  7.6.2 Navigation Implementation ........................................ 189
  7.6.3 Comparison with Spline SLAM ..................................... 193
  7.6.4 Terrain Based Water Edge TAN .................................... 194
7.7 Summary ................................................................. 195

8 Combining Fusion Methods and Real-Time Analysis ..................... 197
  8.1 Solution Accuracy Comparison ....................................... 198
  8.2 Real-Time Implementations .......................................... 200
    8.2.1 Frame Skipping ..................................................... 200
    8.2.2 Delayed Fusion ..................................................... 202
  8.3 Summary ............................................................... 204

9 Conclusions ............................................................ 206
9.1 System Performance ................................................. 207
  9.1.1 Image Processing .............................................. 207
  9.1.2 Spline Based SLAM ........................................... 207
  9.1.3 Visual Spline-TAN ........................................... 208
  9.1.4 Visual Contour-TAN ........................................... 208
  9.1.5 Real-Time Implementation ................................... 209
9.2 Summary of Contributions ....................................... 210
9.3 Directions for Future Work ....................................... 211
  9.3.1 Image Processing .............................................. 211
  9.3.2 High Level Feature SLAM .................................... 212
  9.3.3 Robust Visual Spline-TAN .................................... 212
  9.3.4 Visual Contour-TAN ........................................... 213
  9.3.5 Real-Time Implementation ................................... 213

10 Appendix ................................................................... 227
  10.1 Parametric Curvature Derivation ............................... 227
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Modern Navigation System GNSS Reliance</td>
<td>4</td>
</tr>
<tr>
<td>1.2</td>
<td>Human Visual Navigation Methods</td>
<td>5</td>
</tr>
<tr>
<td>1.3</td>
<td>Computer Vision Navigation Aid</td>
<td>6</td>
</tr>
<tr>
<td>1.4</td>
<td>JabLab Aerial Test Platform</td>
<td>9</td>
</tr>
<tr>
<td>3.1</td>
<td>Camera Frame Axis Definitions</td>
<td>50</td>
</tr>
<tr>
<td>3.2</td>
<td>Camera Model Point and Gradient Definitions</td>
<td>74</td>
</tr>
<tr>
<td>4.1</td>
<td>Colour Content of Tree and Water Aerial Imagery</td>
<td>87</td>
</tr>
<tr>
<td>4.2</td>
<td>Examples of Aerial Imagery</td>
<td>88</td>
</tr>
<tr>
<td>4.3</td>
<td>$RGB$ vs $Lab$ Colour-space Comparison</td>
<td>90</td>
</tr>
<tr>
<td>4.4</td>
<td>Gabor Filter Response Example</td>
<td>91</td>
</tr>
<tr>
<td>4.5</td>
<td>Image Intensity Over-saturation Example</td>
<td>92</td>
</tr>
<tr>
<td>4.6</td>
<td>Effects of Filtering on Image Watershedding</td>
<td>95</td>
</tr>
<tr>
<td>4.7</td>
<td>Parallel Edge Tolerant Filtering Method</td>
<td>96</td>
</tr>
<tr>
<td>4.8</td>
<td>Image Gradient Magnitude Histogram</td>
<td>97</td>
</tr>
<tr>
<td>4.9</td>
<td>Effects of Thresholding on Image Watershedding</td>
<td>98</td>
</tr>
<tr>
<td>4.10</td>
<td>Adjacent Region Colour Similarity Analysis</td>
<td>99</td>
</tr>
<tr>
<td>4.11</td>
<td>Similarity Based Region Merging Example</td>
<td>99</td>
</tr>
<tr>
<td>4.12</td>
<td>Texture Based Region Merging Example</td>
<td>100</td>
</tr>
<tr>
<td>4.13</td>
<td>Edge Extraction From Segmented Imagery</td>
<td>101</td>
</tr>
<tr>
<td>4.14</td>
<td>Edge Gradient and Boundary Texture Extraction</td>
<td>101</td>
</tr>
</tbody>
</table>
7.9 Visual Odometry Innovation Definition ........................................... 184
7.10 Terrain Based Water Edge Estimate Derivation .......................... 185
7.11 Point-Spline Water Edge Estimate Innovations ......................... 186
7.12 TWE Edge Curvature Variance .................................................. 186
7.13 TWE Edge Innovation Variance ................................................. 187
7.14 Mapped Terrain Profile Estimate Result .................................... 188
7.15 Terrain Profile Estimate Accuracy Analysis ............................... 189
7.16 C-TAN Innovation Accuracy ...................................................... 189
7.17 Visual Odometry Innovation Accuracy ......................................... 190
7.18 C-TAN Trajectory Comparison .................................................. 190
7.19 Navigation Accuracy of C-TAN .................................................. 191
7.20 Velocity Estimate Accuracy of C-TAN ......................................... 192
7.21 Computation Times of C-TAN .................................................... 193
7.22 Computation Time Breakdown of Contour TAN ......................... 193
7.23 Navigation Accuracy of C-TAN versus Spline SLAM .................. 194
7.24 Navigation Accuracy of Spline SLAM with C-TAN, versus Spline TAN 195

8.1 Comparison of Presented Visual Navigation Aid Accuracies .......... 199
8.2 Frame-Skipping Methodology for Real-Time Implementation ........ 201
8.3 Navigation Accuracy of Real-Time Spline TAN-SLAM .................. 202
8.4 Navigation Accuracy Loss from Skipping Frames ....................... 203
8.5 Delayed Fusion Methodology with Forward Predictions ............... 204
8.6 Delayed Fusion Navigational Accuracy Comparison .................. 204
List of Tables

1.1 Test Platform Sensor Suite Outline ........................................... 10
1.2 Vehicle Sensor Characteristics .................................................. 10

2.1 Commercial GPS Accuracies ...................................................... 16
2.2 GNSS-RTK Accuracy ................................................................. 16

4.1 Texture Segmentation Classification Matrix .................................... 106
4.2 Simplified Texture Segmentation Classification Matrix .................... 106
4.3 Texture Segmentation Computation Time Breakdown ......................... 107

8.1 Mean Errors of Different Fusion Systems .................................... 200
# List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>Two Dimensional</td>
</tr>
<tr>
<td>3D</td>
<td>Three Dimensional</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>CSS</td>
<td>Curvature Scale Space</td>
</tr>
<tr>
<td>DCM</td>
<td>Direction Cosine Matrix</td>
</tr>
<tr>
<td>DTEM</td>
<td>Digital Terrain Elevation Map</td>
</tr>
<tr>
<td>DGPS</td>
<td>Differential GPS</td>
</tr>
<tr>
<td>DME</td>
<td>Distance Measuring Equipment</td>
</tr>
<tr>
<td>DRS</td>
<td>Dead-Reckoned Solution</td>
</tr>
<tr>
<td>ECEF</td>
<td>Earth Centred, Earth Fixed</td>
</tr>
<tr>
<td>EKF</td>
<td>Extended Kalman Filter</td>
</tr>
<tr>
<td>EM</td>
<td>Electro-Magnetic</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>GIS</td>
<td>Geographic Information System</td>
</tr>
<tr>
<td>GLS</td>
<td>Generalised Least Squares</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>GPGPU</td>
<td>General-Purpose computing on GPU hardware</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>HSV</td>
<td>Hue, Saturation, Value</td>
</tr>
<tr>
<td>IFR</td>
<td>Instrument Flight Rules</td>
</tr>
<tr>
<td>IMU</td>
<td>Inertial Measurement Unit</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>INS</td>
<td>Inertial Navigation System</td>
</tr>
<tr>
<td>Lab</td>
<td>Luminosity, colour a, colour b</td>
</tr>
<tr>
<td>LLA</td>
<td>Latitude, Longitude, Altitude</td>
</tr>
<tr>
<td>LVLH</td>
<td>Local Vertical, Local Horizontal</td>
</tr>
<tr>
<td>MAV</td>
<td>Micro Air Vehicle</td>
</tr>
<tr>
<td>MTOW</td>
<td>Maximum Take-Off Weight</td>
</tr>
<tr>
<td>NED</td>
<td>North, East, Down</td>
</tr>
<tr>
<td>NLLS</td>
<td>Non-Linear Least Squares</td>
</tr>
<tr>
<td>OLS</td>
<td>Ordinary Least Squares</td>
</tr>
<tr>
<td>RGB</td>
<td>Red, Green, Blue</td>
</tr>
<tr>
<td>RSS</td>
<td>Root Sum Square</td>
</tr>
<tr>
<td>RTK</td>
<td>Real Time Kinematic</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>SRTM</td>
<td>Shuttle Radar Topography Mission</td>
</tr>
<tr>
<td>SWBD</td>
<td>SRTM Water Body Data</td>
</tr>
<tr>
<td>TACAN</td>
<td>Tactical Air Navigation System</td>
</tr>
<tr>
<td>TAN</td>
<td>Terrain Aided Navigation</td>
</tr>
<tr>
<td>TANS</td>
<td>Terrain Aided Navigation System</td>
</tr>
<tr>
<td>TERCOM</td>
<td>Terrain Contour Matching</td>
</tr>
<tr>
<td>TERPROM</td>
<td>Terrain Profile Matching</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aerial Vehicle</td>
</tr>
<tr>
<td>UKF</td>
<td>Unscented Kalman Filter</td>
</tr>
<tr>
<td>USYD</td>
<td>The University of Sydney</td>
</tr>
<tr>
<td>VFR</td>
<td>Visual Flight Rules</td>
</tr>
<tr>
<td>VHAD</td>
<td>Visual Horizon Attitude Determination</td>
</tr>
<tr>
<td>VHF</td>
<td>Very High Frequency</td>
</tr>
<tr>
<td>VOR</td>
<td>VHF Omni-directional Radio Range</td>
</tr>
<tr>
<td>WGS</td>
<td>World Geodetic System</td>
</tr>
<tr>
<td>WGS84</td>
<td>World Geodetic System, 1984</td>
</tr>
<tr>
<td>WLS</td>
<td>Weighted Least Squares</td>
</tr>
</tbody>
</table>
List of Abbreviations

6-DOF Six Degrees of Freedom
BS-SLAM Basic Spline SLAM
B-Spline Basic Spline
C-TAN Contour-based Terrain Aided Navigation
GPS/INS GPS-assisted Inertial Navigation System
TAN-SLAM Terrain Aided Navigation, Coupled With SLAM
VOR/DME Combined VOR & DME Localisation Beacons

Units of Measurement

m Metres
km Kilometres
s Seconds
ms Milliseconds
ms\(^{-1}\) Metres per Second
kg Kilograms
Hz Hertz
ft Feet
kts Knots
Mathematical Notation

- $x$: Scalar
- $\vec{x}$: Vector
- $X$: Matrix
- $\bar{x}$: Mean
- $\dot{x}$: Time Derivative
- $x_i$: $i$-th Element of Vector $x$
- $X_{i,j}$: $i, j$-th Element of Matrix $X$
- $x[i]$: $i$-th Iteration of Vector $x$
- $F(a, b)$: Function $F$, with inputs $a, b$
- $\lfloor x \rfloor$: Rounding of $x$ to Nearest Integer
- $\lfloor x \rfloor$: Floor Rounding of $x$
- $\lceil x \rceil$: Ceiling Rounding of $x$
- $|x|$: Absolute Value of $x$
- $X^T$: Transpose of Matrix $X$
- $X^{-1}$: Inverse of Matrix $X$
- $C_x$: DCM Rotation about Dimension $x$
- $C_{vu}$: DCM Rotation Matrix from $u$ to $v$ Axes
List of Symbols

\( \sigma \)  Standard Deviation
\( \chi^2 \)  Chi-Squared Distribution
\( I \)  Identity Matrix
\( C_{bn} \)  DCM Rotation Matrix from Earth to Body Axes
\( C_{cb} \)  DCM Rotation Matrix from Body to Camera Axes
\( \phi \)  Euler Roll Angle
\( \theta \)  Euler Pitch Angle
\( \psi \)  Euler Yaw Angle
\( R \)  Camera Frame Range to Point
\( \chi \)  Camera Frame Bearing to Point
\( \lambda \)  Camera Frame Inclination to Point
\( \gamma \)  Camera Frame Curve Rotation, Parallel to \( R \) Vector
\( \epsilon \)  Camera Frame Curve Rotation, Perpendicular to \( R \) Vector
\( p \)  Body Roll Rotation Rate
\( q \)  Body Pitch Rotation Rate
\( r \)  Body Yaw Rotation Rate
\( a_x \)  Body Forwards Acceleration
\( a_y \)  Body Sideways Acceleration
\( a_z \)  Body Vertical Acceleration
\( q_0, q_1, q_2, q_3 \)  Quaternion Attitude Representations
\( x_e \)  Distance North from Origin
\( y_e \)  Distance East from Origin
\( z_e \)  Distance Down from Origin
\( g \)  Acceleration due to Gravity
\( X \)  Kalman State Vector
\( P \)  Variance of Kalman States
\( Q \)  Variance of Process Measurements
\( R \)  Variance of Sensor Measurement
\( Y \)  Point Co-ordinates in Camera Frame
\( H \)  Point Co-ordinates in Earth Frame
Chapter 1

Introduction

1.1 Background

Vision is undoubtedly the most dominant sense that most animals, including humans, use. We use it constantly to avoid danger, find things we need or want, navigate to where we want to be, and collect new information. In an aeronautical setting, visual navigation techniques such as visual flight rules (VFR) are the simplest methods. Furthermore, visual navigation techniques employed by a pilot require no equipment, which potentially could malfunction. They are also perhaps the most reliable techniques for collision avoidance due to this lack of dependence on extra equipment.

Despite the obvious advantages demonstrated by human visual navigation systems, modern aerial navigation generally relies on a Global Navigation Satellite System (GNSS) to localise airborne vehicles. This approach is inherently risky, as there are a number of reasons [4–6] why GNSS localisation can fail. Should this occur, vehicle pose estimation must rely on inertial navigation systems. As inertial measurements detect accelerations, this results in position estimates that diverge at a geometric rate over time. For inertial measurement units (IMU) likely to be fitted to light aircraft or unmanned vehicles, position uncertainties can drift to many kilometres in the time span of minutes. Reasons GNSS localisation can fail include, but are not limited to, obstruction of satellite line of sight or multi-path signal errors from reflections in urban canyons. Other situations arise from operations in shielded environments, the presence of excessive electromagnetic interference, as well as atmospheric or solar activity.
GNSS failure tolerance is especially pertinent in military environments, where active and intentional jamming or spoofing of radio signals is possible, and entirely likely, even when facing low-tech adversaries. Furthermore, when facing technologically advanced combatants, GNSS satellite destruction is a possibility, or even nuclear detonations resulting in ionospheric disturbances. Finally, reliance on GNSS for aerial navigation results in complete dependence on the foreign powers that operate these systems, i.e. the USA (GPS), or alternative systems such as Russia (GLONASS), etc. There is no guarantee that these foreign powers will continue to operate these services, especially if they deem it to no longer be in their best interests to do so. The ESA's new, civilian-controlled global navigation satellite system (GNSS) Galileo, and any future civilian systems, help to limit the possibility of switch-off, however cannot eliminate it completely. For these reasons, development of navigation systems which do not require the use of a GNSS for operation are an active area of scrutiny in modern research fields.

Perhaps the most obvious solution to the GNSS-denied navigation problem is to task a human pilot with this work. This solution however is problematic in many situations, as humans are heavy, as are the support systems which they require. This results in large amounts of extra aerial platform weight, volume and cost, which can be prohibitive for many missions. Furthermore, the use of a human pilot can be a serious drawback for a number of other reasons. These include safety, where using human pilots would be impractical or impossible for many military applications, as well as some civilian situations. Others include restraints on mission time due to concentration and sleep requirements, human acceleration limitations, as well as any essential thermal or radiation shielding. Strategic limitations also arise from the presence of a human aboard a vehicle, as the aerial platform can no longer be considered expendable. The necessity in having the vehicle return intact, and keeping the human safe, can seriously limit actions the vehicle could otherwise perform without compromise.

Human visual navigation can be performed without a human pilot physically aboard the vehicle using remote piloting systems, which bypass most of the drawbacks of onboard human piloting. These systems do however come with their own limitations, such as requiring line-of-sight between vehicle and pilot, or potentially complex and widespread permanent communications infrastructure. An alternative to this is to use a dedicated satellite link, however these are expensive. Finally, any remote pilot system is potentially just as susceptible to signal loss, jamming or spoofing as the GNSS navigation systems they replace.

A second, obvious solution to the GNSS-denied navigation problem is to employ the use of a high
precision inertial measurement unit. These systems do exist, however their weight, size and cost increase rapidly with increasing accuracy. Furthermore, these are still only a temporary solution, as they still have the fundamental inertial navigation weakness of geometric error accumulation. A higher accuracy IMU simply slows the accrual of drift, and therefore delays the point where position error becomes too large to manage. Therefore even an extremely heavy and expensive IMU of the highest accuracy cannot be used for extended navigation without some localisation assistance.

Onboard active radar mapping systems can be used for autonomous vehicle localisation, such as the Terrain Contour Matching (TERCOM) system [7, 8]. These systems use a scanning radar rangefinder to generate an estimate of the shape of underlying terrain, which is then matched to an onboard database. The critical drawback of this system however, is due to the active radar system. These systems are heavy and consume a large amount of power. This therefore places limits on the size of the platform it can be fitted to. Furthermore, in military situations this radar range finder also acts as a clear beacon to any radar detection apparatus, preventing any stealthy operations from being conducted.

For these outlined reasons, any aerial mission which required unmanned operation would benefit from a vision based navigation system. Such a navigation system would be independent of GNSS, providing the ability to operate in GNSS denied environments. Furthermore, as it could be modelled on methods a human navigator would use, it would consist of only passive sensors, hence not interfering with any stealth properties of a vehicle. Identification and localisation via human-recognisable features would allow inertial drift to be eliminated in known environments. In addition, relative navigation techniques from detected features allows drift to be limited even in uncharacterised environments. Finally, such a system would operate using digital video cameras, which exhibit low weight, low power draw, and are inexpensive. These costs would be associated with the processing of imagery and data fusion, instead of the operation of active sensors.

Relative navigation systems are the main focus of the Simultaneous Localisation and Mapping (SLAM) algorithm presented by Cheeseman et. al. [9]. This influential work is an augmentation to the extended Kalman filter, which allows the stochastic time varying parameter estimator to initialise, refine and discard parameters during operation. This enables the filter to gauge the locations and characteristics of previously unknown visible features, while simultaneously estimating the vehicle pose. A further benefit of this system is that if uniquely identifiable and repeatable features are not viewed, the system
reverts to the best available information, i.e. inertial integration. When visually distinct features return, cross coupling between different states can result in a reduction of accrued drift during this period of dead-reckoning. SLAM also provides the possibility for loop closure, where re-visiting previously viewed features can both eliminate accrued drift since the feature was last seen, as well as improve the estimate uncertainty of other viewed features. This is a significant benefit of SLAM over other methods of visual odometry, such as optical flow techniques.

1.2 Problem Statement

Any aerial navigation system necessitates a highly accurate, instantaneous localisation estimate. This is required to ensure safe and precise guidance and control. Modern navigation aids generally rely on inertial information, to provide a continuous localisation solution. The integration of inertial measurements results in unavoidable dead-reckoning drift, due to sensor noise and/or biases. Modern navigation systems therefore, employ the use of GNSS position readings to negate this drift (figure 1.1, part a). This makes the system totally reliant on GNSS position updates for extended navigation.

![Figure 1.1: Modern navigation systems rely completely on GNSS systems (part a). If these signals are unavailable, rapid localisation divergence results (part b).](image)

This reliance on GNSS is highly risky, as satellite signal reception can fail for a wide range of reasons. These include unintentional factors, such as atmospheric or solar disturbances, EM (electro-magnetic) interference, EM shielded environments and multi-path signal reflections. Intentional failure sources also
exist, such as jamming, signal spoofing and satellite destruction. The result of any of these occurrences would culminate in a rapid divergence of the system localisation estimate (figure 1.1, part b).

Figure 1.2: A human pilot can navigate an aerial vehicle using visually distinct features, and a map.

Before the deployment of GPS, and other GNSS variations, radar-based terrain contour matching (TERCOM), was employed to provide position updates, limiting inertial drift for unmanned vehicles. This navigation aid, however, requires the use of a cumbersome active radar system, which is easily detected by potentially hostile forces. Ground-based radio beacons have also been implemented as navigation aids, however, these systems are also subject to jamming, just like GNSS. For these reasons, robust tolerance to GNSS failure is predominantly provided by a human navigator, or pilot. As a human pilot can localise a vehicle exclusively using visual information and a map (figure 1.2), no external infrastructure is required. Despite this advantage, this solution is problematic, for a wide range of different strategic, performance, cost and mission-based limitations.

By utilising video image processing, a computer-based navigation system can be developed and implemented. This would resolve aerial navigational solutions, by utilising similar methods to a human. Emulation of human navigation techniques provides an established and validated process for successful localisation and tracking, also allowing easy human oversight if required. This provides tolerance to GNSS signal loss, as satellite communications may be replaced by computer-vision feature associations (figure 1.3).

A computer-based visual navigation system would have a number of advantages over other navigation systems. Firstly, as GNSS is not required, aerial vehicles could operate in environments where radio signals cannot be relied upon. The removal of any on-board humans eliminates a number of undesirable
restrictions, such as platform size, mass and performance, as well as dismissing a number of possible mission limitations. Finally, the exclusive use of passive sensors enables this system to be utilised for combat or hostile surveillance missions, where stealth is vital.

![Computer-vision based feature associations can replace GNSS navigation position updates.](image)

This thesis will outline the development of computer-vision techniques, which are demonstrated to provide the necessary localisation solutions required to navigate an aerial vehicle. This system operates without the use of GNSS, or any active sensors. In this thesis, edge curves are analysed as potential features for a terrain aided navigation system (TANS). Vision-based terrain contour profile matching is also investigated as a navigation aid. The navigational accuracy and computational performance of these systems are discussed, and finally, a real-time visual TANS solution is presented, and examined.

The visual navigation system outlined in this thesis will consist of a number of different modules, which operate based on the availability of information:

- Inertial-based state predictions allow a continuous vehicle state estimate, even in the absence of any external information. This solution may be assisted by typical, robust aerial sensors such as a magnetometer and static pressure measurements.

- Colour video cameras mounted to this aerial vehicle are used to capture visual information. This aerial imagery is processed in order to extract any human-recognisable edge features.

- In the presence of visually distinct edge features spline-based SLAM is employed to limit inertial drift, and generate an edge curve feature map of the surrounding terrain.
• Geo-referenced aerial and satellite imagery sources can be used to create a known feature map database. This map will be of the same form as that generated by the spline-SLAM module, and contain the same types of features.

• During aerial operations, any mapped features may be compared to this curve database, and associations made. Any matches can be used to bound position drift with a TAN update. This will also help to constrain the SLAM feature map.

• Optical flow information obtained from video imagery can be used to infer the profile shape of surrounding terrain. This can be compared with a known DTEM source, via a TERCOM-like process, resulting in a passive, visual, contour-based TANS. Optical flow information may also be used to perform visual odometry.

Previous analytical investigations of SLAM [10] prove that a small number of high-quality, robust innovations is beneficial for both navigation accuracy and computational expense. It is therefore important to analyse the available information at any particular moment for the robustness and innovation variance of information available from these outlined sources. Algorithm performance may therefore be improved by switching between these different sources, based on the best available data available at any time.

1.3 Contributions

A summary of the outcomes of this thesis are outlined below:

1. Development of a visual navigation aid capable of operating either with, or without, a-priori feature database information. This aid should also rely on high level, human identifiable, edge features.

2. Investigation of a texture segmentation based method to determine edge features present in a camera frame, with a system to extract these edges for classification. Focus in this thesis is towards riverine edges between water and bushland.

3. Development of a monocular, 6-DOF spline based SLAM algorithm which assists navigation through the tracking of detected edge features.
4. Derivation of a spline parametric re-weighting method which eliminates sensitivity issues caused by end point clamping.

5. Investigation of methods to generate a data-compact, high level feature database from existing aerial and satellite imagery sources. Texture segmentation and edge detection techniques are used to extract this information.

6. Development of data association techniques which allow matching of viewed edge features to corresponding database features. This can then be used to perform terrain aided navigation, allowing absolute position solutions to be obtained exclusively from visual information.

7. Derivation of a ray-tracing based range-to-terrain estimator, allowing the use of a DTEM to restrict initialisation problems inherent in bearing-only SLAM.

8. Investigation of an optical flow based method for estimating the height profile of underlying terrain, operating exclusively using passive, visual measurements.

9. Development of a data association technique for matching optical flow based terrain profiles with a known DTEM profile, allowing absolute position solutions to be obtained.

10. Investigation of a visual odometry implementation which operates in parallel to the visual terrain contour matching system, limiting inertial drift of velocity estimates.

11. System validation using real flight-test data obtained using the University of Sydney Jabiru J400 aerial test platform.

12. Investigation of a technique for optimising the use of visual localisation updates which are delayed due to processing time.

### 1.4 Test Platform

For reasons of algorithm validation, the flight data analysed in this paper was obtained using the University of Sydney (USYD) J400 Jabiru light aircraft (figure 1.4). This is a 4-seat experimental aerial vehicle platform with a maximum take-off weight of 700 kg. It has a flight ceiling of 15,000 ft and a cruise speed
of 120 kts. This vehicle is used for training and research purposes at the University of Sydney, and as such is fitted with custom data collection hardware and software.

Figure 1.4: The USYD Jabiru J400 experimental aerial test platform.

The test platform has been modified to include a high-precision 6-axis laser ring IMU, which outputs raw inertial measurements. This information is combined with three GPS receivers, allowing accurate position localisation, as well as differential carrier phase attitude determination. Raw body-frame measurements of the Earth magnetic field vector are also recorded using a digital magnetometer, supplying alternate attitude information. An air data probe is also fitted to the vehicle, providing both airspeed and static pressure measurements. This air data probe also outputs angle of attack and sideslip measurements, while control position transducers record all control deflections during flight.

The visual data systems consist of seven individual digital video cameras, two of which are high resolution cameras facing down and diagonally forward (in the body frame of reference). These are used for recording navigation information for feature detection. The other five cameras are mounted facing different lateral directions, and are used for horizon profile detection.

Four dedicated Mac Mini compact computers are used to collect and record the data provided by all of these sensors. The first computer records the flight data, while the second records the five horizon cameras. The final two computers record the two higher resolution navigation cameras.

The GPS/INS solution provided by the NovAtel SPAN system is used to provide a truth estimate, and is therefore used to judge accuracy of the visual navigation systems outlined in this thesis. Once algorithm validation is achieved, the system could be operated equivalently on any autonomous UAV,
guided missile, or autopilot system with a similar sensor suite. These sensors are outlined in table 1.1.

<table>
<thead>
<tr>
<th>Count</th>
<th>Sensor</th>
<th>Type</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NovAtel ProPak-G2plus</td>
<td>GPS/INS</td>
<td>100 Hz</td>
</tr>
<tr>
<td>1</td>
<td>NovAtel HG1700 SPAN</td>
<td>IMU</td>
<td>100 Hz</td>
</tr>
<tr>
<td>3</td>
<td>CMC Allstar 12</td>
<td>GPS Receiver</td>
<td>10 Hz</td>
</tr>
<tr>
<td>1</td>
<td>Honeywell HMR2300</td>
<td>Magnetometer</td>
<td>100 Hz</td>
</tr>
<tr>
<td>1</td>
<td>SpaceAge Control 100400</td>
<td>Air Data Probe</td>
<td>500 Hz</td>
</tr>
<tr>
<td></td>
<td>SpaceAge Control</td>
<td>Position Transducer</td>
<td>500 Hz</td>
</tr>
<tr>
<td>5</td>
<td>Unibrain Fire-i</td>
<td>640x480 Camera</td>
<td>30 Hz</td>
</tr>
<tr>
<td>2</td>
<td>Prosilica GC-1020C Gig-E</td>
<td>1024x768 Camera</td>
<td>25 Hz</td>
</tr>
</tbody>
</table>

Table 1.1: Test Platform Sensor Suite Outline

The vehicle system described in this paper is based around a 10–state Extended Kalman Filter, with three body–frame velocity components \([u, v, w]\), four quaternion attitude components \([q_0, q_1, q_2, q_3]\), and three world–frame position components, \([x_e, y_e, z_e]\). A dead–reckoned solution (DRS) can be determined from the measurements provided by a 6–axis IMU. This provides three acceleration components \([a_x, a_y, a_z]\), plus roll, pitch and yaw rotation rate components \([p, q, r]\), all in the body frame. In order to limit attitude drift, a magnetometer sensor can be used to fuse pose data. The magnetometer and static pressure measurements are also fused into the dead–reckoned IMU integration solution. This provision allows for a more authentic GNSS–denied trajectory estimate with which to compare the visual navigation system outlined in this paper. The accuracy parameters of the IMU and digital camera are outlined in table 1.2.

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerometer</td>
<td>Update Rate</td>
<td>100 Hz</td>
</tr>
<tr>
<td></td>
<td>(a_x) noise ((\sigma))</td>
<td>0.0198 (g)</td>
</tr>
<tr>
<td></td>
<td>(a_y) noise ((\sigma))</td>
<td>0.0318 (g)</td>
</tr>
<tr>
<td></td>
<td>(a_z) noise ((\sigma))</td>
<td>0.0938 (g)</td>
</tr>
<tr>
<td>Gyroscope</td>
<td>(p) noise ((\sigma))</td>
<td>0.549° (s^{-1})</td>
</tr>
<tr>
<td></td>
<td>(q) noise ((\sigma))</td>
<td>0.297° (s^{-1})</td>
</tr>
<tr>
<td></td>
<td>(r) noise ((\sigma))</td>
<td>0.375° (s^{-1})</td>
</tr>
<tr>
<td>Camera</td>
<td>Update Rate</td>
<td>5 Hz</td>
</tr>
<tr>
<td></td>
<td>FOV</td>
<td>45.25°</td>
</tr>
<tr>
<td></td>
<td>Resolution</td>
<td>1024 x 768</td>
</tr>
<tr>
<td></td>
<td>Camera Offset</td>
<td>([-2.5^\circ, -90^\circ, 0.0^\circ])</td>
</tr>
</tbody>
</table>

Table 1.2: Vehicle Sensor Characteristics
1.5 Chapter Outline

Chapter 1 introduces the thesis, defines the problems which this thesis targets and presents outcomes.

Chapter 2 is a literature review of past research work in the areas of texture segmentation, GNSS-denied aerial navigation, computer vision navigation, as well as curve based localisation and mapping. Observations regarding relevant sections of these studies are noted, along with limitations and areas of general improvements when pertinent to the thesis goals.

Chapter 3 analyses the background knowledge and theoretical work directly related to this thesis. This includes the definitions of reference frames and coordinate transforms, as well as analytical camera model analysis. This chapter also outlines the basic spline, and statistical based algorithms of data fusion and parameter estimation.

Chapter 4 develops a near real-time texture segmentation algorithm for separating areas in a camera image of different ground coverage types. This algorithm uses both colour and texture information, enhancing robustness to variations in colour properties of ground cover types. Methods are outlined to improve tolerance of lighting conditions and problems arising from noise, or complex frames. This segmentation is used to designate edge features present in viewed terrain, which are later used as parameter estimation measurements.

Chapter 5 outlines a monocular spline-based extended Kalman filter based simultaneous localisation and mapping system, for improving pose certainty during GNSS-denied aerial navigation. This provides the full 6-DOF parameter estimation of the vehicle pose. Analytical sensitivity matrices of both camera transformations and spline solutions are developed, greatly improving the computation time of the Kalman filter and allowing real-time operation of the filter. Post-processed experimental data of a test flight is used to demonstrate performance and accuracy.

Chapter 6 introduces the use of a known edge feature database to provide visual position updates to the Kalman filter, which can be used to eliminate dead reckoning drift. Automated creation of this database
through the analysis of Google Earth imagery is presented, which is of benefit due to the availability of satellite and aerial reference imagery for the majority of the Earth’s landmass. Furthermore, the image processing presented in this chapter allows only the useful features of this imagery to be extracted, significantly reducing the amount of data to store relative to the raw images.

Chapter 7 presents the use of optical flow information retrieved through processing of consecutive image frames to infer the range to viewed terrain. This estimate can be used both for mapping the expected terrain height of surveyed ground, and as a method of constraining dead reckoning drift. This drift reduction is performed through terrain profile matching to a known database. Optical flow is also used for visual odometry. This chapter also analyses how the shape of terrain affects the movement of water edges. This information can be used to adjust the locations of known database water edge features should water level changes be known. Similarly, estimating changes in the position and shape of water edge features can help to estimate any changes in water level of a lake, river or dam body.

Chapter 8 analyses the relative performances of the different presented visual navigation methodologies, as well as the performance and accuracy of a combined approach.

Chapter 9 is the conclusion of the thesis, where the outcomes of this work are summarised and areas of future work outlined. The implications and performance of this work is discussed.
Chapter 2

Related Work

2.1 Introduction

This chapter provides an extensive literature review of the current status of aerial navigation systems, along with other areas relevant to the development of this thesis. The relative advantages and limitations of particular avenues of research are discussed, as well as observations identifying areas of possible improvement, especially related to the goal of this thesis. This chapter has been separated into a number of distinct sections, which are outlined below:

Section 2.2: Commercial Aerial Navigation Aids  The current state of typical commercial aerial navigation systems and aids available to most airborne platforms, is reviewed and discussed in this section. These include ground based radio beacon systems, as well as GNSS infrastructure and receivers. This section also reviews the typical accuracies of basic and more advanced GNSS systems.

Section 2.3: Sensors & Data Fusion  The advantages and shortcomings of typical sensors fitted to aerial platforms are discussed, as are sensor fusion algorithms currently popular in research fields. These fusion techniques include the popular Kalman filter, as well as more recent extensions and generalisations which improve its performance. SLAM methodology, as well as the logical extension of terrain aided SLAM, are also discussed.
Section 2.4: Contour Based Aerial Navigation Methods  Current implementations of navigation systems, which operate through estimation and matching of terrain height profiles, are discussed in this section. Present research into passive, visual-only alternatives are also reviewed. This includes both the use of stereo camera range estimates, as well as inferred range from optical flow relative to vehicle motion. The use of horizon profiles to perform localisation is also examined.

Section 2.5: Feature Detection  The features used for current aerial and ground based SLAM implementations are explored in this section, involving observations on the relative benefits and drawbacks of abstract features such as the SURF. This section also outlines methods of detecting higher order curve features, through an overview of edge detection techniques, along with the use of texture segmentation.

Section 2.6: Feature Based Aerial Navigation Techniques  This section covers the use of visual features to limit drift in unknown environments using SLAM. Also discussed are current implementations of visual terrain aided navigation techniques and the features used in this process. Examples discussed involve the use of road bridges and road intersections, with the benefits and drawbacks of the use of such features considered.

Section 2.7: Data Association  Current research towards methods necessary for the use of higher-level, human recognisable features are investigated here. This focusses on methods by which data association can be performed on curved feature representations, such as basic splines. Focus towards tolerance to affine transformations, occlusions and inclusions are also given consideration.

Section 2.8: Summary  The final section to this chapter is a summary of the important observations which directly relate to the development of this thesis. It also serves as a review of the required extensions to previous works necessary.
2.2 Commercial Aerial Navigation Aids

There exists a number of commercial navigation and localisation systems for aerial platforms, which either act as an aid for human pilots, or which provide position information for computer based navigation systems. These systems almost exclusively operate using ground or satellite-based radio frequency electromagnetic emissions. The benefits of such methods involve comparatively low on-board power requirements, and favourable transmission properties through the atmosphere. These commercial systems exhibit a wide range of accuracies based on the systems they use for localisation, and the quality of the sensors they contain. All of these systems rely on external infrastructure. Other systems which do not rely on external infrastructure generally use active radar sensors such as TERCOM [7, 8].

2.2.1 Ground Based Radio Systems

The distance measuring equipment (DME) system is a ground based radio transmitter and receiver which assists aerial vehicle localisation. This is performed by the emission of radio pulses from a transmitter on the aircraft. These pulses are detected by the DME ground station and quickly re-emitted. Once these returned pulses are detected by the air vehicle, the time taken for the combined return trip can be evaluated and used to estimate distance from the ground station. A parallel localisation aid known as VHF omni-directional radio range (VOR) enables the calculation of relative bearing to a ground station. The VOR ground based beacon emits radio using a narrow focus transmitter, which rapidly sweeps a $360^\circ$ arc around the station. This is coupled with a secondary, omni-directional radio beacon, which pulses with each rotation of the focussed transmitter. The time interval between detection and phase differences in the signals, from both these transmitters, can be used to calculate the relative bearing of the vehicle from the ground station.

In order to fully localise a vehicle, three DME beacons must be viewed simultaneously. A position solution can also be obtained through the reception of two simultaneous VOR beacons. Alternatively, a single ground station operating both DME and VOR signals can be used for a definitive position fix. A more modern, military alternative to the VOR/DME system is known as the tactical air navigation system or TACAN, which exhibits greater accuracy, however still operates using the same principles.
The development and deployment of ground based radar systems predominantly occurred during the early to mid 20th century. These systems have mainly been superseded through the deployment of the satellite based GPS in the late 20th century, and other GNSS networks. The advantages of these systems is a significantly improved cost/coverage ratio. Although satellite systems are highly expensive to deploy, their high altitude and orbital nature result in large service coverage. In contrast, single ground based installations are comparatively cheap, but exhibit much shorter range due to low altitude and more pronounced atmospheric signal distortion effects. A large number of ground based transmitters would therefore be required to service any reasonable area.

### 2.2.2 Space Based Radio Systems

Commercial GNSS systems exhibit accuracies ranging from the comparatively poor performance of standard GPS, to significantly improved systems such as differential GPS (DGPS) and real time kinematic GPS (RTK). The typical accuracies of these systems is shown in table 2.1 [11].

<table>
<thead>
<tr>
<th>GPS Type</th>
<th>Standard</th>
<th>DGPS</th>
<th>RTK-GPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal Position (m)</td>
<td>30</td>
<td>2</td>
<td>0.2</td>
</tr>
<tr>
<td>Vertical Position  (m)</td>
<td>40</td>
<td>5</td>
<td>0.2</td>
</tr>
<tr>
<td>Horizontal Velocity (m/s)</td>
<td>0.13</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>Vertical Velocity   (m/s)</td>
<td>0.16</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 2.1: Comparison of commercial GPS accuracies (95% certainty) [11]

The fusion of high quality inertial measurements from extremely accurate IMU systems can help provide temporary tolerance to the loss of satellite communication in GNSS systems. The performance of such a high precision system [12] is shown in table 2.2, both when satellite signals are available, as well as after 60 seconds of GNSS-denied navigation.

<table>
<thead>
<tr>
<th></th>
<th>GNSS-RTK</th>
<th>60s Signal Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal Position  (m)</td>
<td>0.02</td>
<td>5.71</td>
</tr>
<tr>
<td>Vertical Position    (m)</td>
<td>0.05</td>
<td>1.60</td>
</tr>
<tr>
<td>Horizontal Velocity (m/s)</td>
<td>0.02</td>
<td>0.212</td>
</tr>
<tr>
<td>Vertical Velocity    (m/s)</td>
<td>0.01</td>
<td>0.059</td>
</tr>
</tbody>
</table>

Table 2.2: Accuracy of high precision GNSS-RTK system [12]

Clearly the loss of contact with GNSS transmissions causes the localisation estimate of the vehicle to degrade over time. As using inertial information to estimate changes in position relies on a double
integration of sensor measurements, it is expected that the position solution will continue to degrade at a quadratic rate. This shows that even the highest precision inertial sensors available for small aerial vehicles, cannot provide accurate localisation solutions in GNSS denied environments for more than a few minutes.

The Russian GNSS service, known as GLONASS, provides similar performance to the USA’s GPS. Typical horizontal accuracies are 21.2 m, with expected vertical position errors of 39.1 m, with 95% confidence [13]. More recent areas of research involve combining the use of the different GNSS services, including GPS, GLONASS and Galileo. Not only does this increase the accuracy of localisations, the reception of a larger number of satellite signals improves tolerances to partial line-of-sight obstructions. Therefore, combined GNSS solutions are more tolerant to indoor, or urban-canyon scenarios [14]. Regional GNSS services such as BeiDou (China) may also help to improve accuracy and reliability [15].

2.2.3 Observations

As digital video camera sensors are significantly less expensive than high precision inertial sensors, a visual localisation system with similar performance to a GNSS system would be highly beneficial to the aeronautical industry. Such a system would allow navigation without reliance on any external infrastructure, while retaining passive operation.

2.3 Sensors & Data Fusion

2.3.1 Commonly Available Sensors

Most aerial vehicles have a typical suite of sensors available to them, which provide aircraft state information to a human pilot or a computer guidance system.

The inertial measurement unit (IMU) is a fully self-contained sensor, which collects acceleration data related to the vehicle. An IMU system often contains three sensors, enabling acceleration estimates to be made in any dimension. More advanced systems also include three gyroscope sensors, with which to determine rotation rates. These sensors are specifically used for computer navigation systems. Human
pilots tend to use their inner ear, however may also use a skid-ball indicator to determine sideways accelerations.

The altitude of an aerial vehicle can be determined using a static pressure measurement. This is often presented to a human pilot in a cockpit, via an analogue altimeter. Computer systems will use a transducer to digitise this measurement. Altitude calculations from static pressure measurements are an inherently relative method, as the current pressure at local sea level must be known. This value must be obtained either before flight, or via external communication infrastructure. Any changes in local sea-level air pressure during flight can therefore result in biased altitude measurements.

The velocity of a vehicle is typically determined from a dynamic pressure measurement. This is obtained through the use of a pitot tube, which also takes the static pressure measurements used for altitude determination. The difference in pressure between dynamic and static measurements can be used to determine the relative movement of the aerial vehicle through the air. The drawback of this sensor is that navigation is pertinent to world frame velocities, whereas dynamic pressure resolves velocity relative to the air. Relying on dynamic pressure measurements to estimate velocity will therefore result in biases in the presence of any wind.

Measurements of the Earth’s magnetic field can be used to estimate the attitude of an aerial vehicle. This is performed using a magnetometer, which in the case of a human pilot aid, can simply be a magnet suspended in liquid. Computer navigation systems can employ digital magnetometers, where the magnetic flux is measured in three components, providing a vector representation. This can be used in conjunction with known magnetic data, ascertained at the vehicle’s current location, to help determine the pose of the vehicle. As this sensor measures a single vector quantity, this information alone is not sufficient to fully resolve the attitude of an aerial vehicle. Instead, any attitude solution is free to rotate about the magnetic axis. If other orientation information sources are available, the magnetometer can instead help to constrain the attitude solution.

Definitive localisation measurements can be obtained using GPS, or other GNSS services. These are exclusively computer based systems, involving the reception of radio signals. The receipt of highly precise time-stamped signals allows a GNSS system to triangulate position to a high accuracy. This system operates on very low power, and is also compact and light weight, as it is completely reliant on
external infrastructure, namely orbital satellites. The main drawback of the system is also caused by its predominantly off-board nature, making the system highly sensitive to external effects such as jamming or interference.

A second sensor which obtains definitive position estimates is TERCOM [7, 8], which is a self-contained system operating through radar based scans of underlying terrain. TERCOM is predominantly found on missile systems, with a similar system known as TERPROM [16] used on some other military aerial vehicles. These radar rangefinder scans result in a local terrain profile estimate, which can be matched to a known terrain profile database, localising the vehicle. As it is a self-contained system, operation tends to be robust to external effects such as jamming. This is countered by drawbacks, such as being significantly heavier than GNSS receiver systems, as well as requiring significantly more power to operate the active radar device. Further disadvantages arise due to the visibility of the active radar system, making stealthy operations impossible.

### 2.3.2 Data Fusion Systems

As outlined in section 2.3.1, there exists a number of sensors which are commonly found on aerial vehicles. When employed in computer based navigation systems, these sensors will generally be used in a fusion estimation process. The fusion of different sensor measurements is performed in order to balance the disadvantages of individual sensors with corresponding advantages of others.

A popular and simple fusion system involves the coupling of absolute GNSS localisation solutions with inertial information from an IMU. The benefit of this fusion is that an IMU provides extremely accurate, high frequency measurements, allowing continuous pose estimates over short integration times. The GNSS solutions occur at a much lower frequency, however they directly gauge the position of the vehicle. This means that such measurements do not diverge over time, and can therefore be used to help correct inertial integration drift. Localisation estimates will therefore be bounded, however the accuracy of the system will be dependent on the signal integrity. The GNSS assisted fusion system is therefore sensitive to atmospheric conditions, characteristics of the local environment, as well as system security settings. These factors can still result in significant position measurement errors.
2.3.3 Linear Kalman Filter

The Kalman filter [17] is a least-squares based state estimator, that has been generalised to also predict the values of parameters which change over time. This is performed using a two-step process. The first step involves a prediction of a future state based on known information of the state of the parameters, and also any models describing how the parameters change. An example of this is the motion of a vehicle, where mathematical models and inertial measurements can be used to estimate any future vehicle state. This step also returns an increasing uncertainty of this expected parameter, based on the variance of any process measurements, and of the existing parameter estimates.

The second step of the Kalman filter is the update step. This involves the use of sensors, which either directly or indirectly measure these parameters. This information is used to decrease the uncertainty of the estimates. Examples of measurements include range and bearing to features such as beacons, allowing the calculation of position. Other examples may be of dynamic pressure, resulting in velocity measurements.

The linear Kalman filter is an optimal solution to linear time-varying parameters, where sensitivity matrices involved in the prediction and update steps are invariant. Most real systems however are non-linear, and therefore cannot be accurately modelled using a linear Kalman filter.

2.3.4 Extended Kalman Filter (EKF)

The extended Kalman filter (EKF) is a simple generalisation of the linear Kalman filter for non-linear systems. Instead of being invariant, the sensitivity matrices involved in the prediction and update steps are re-calculated at each time step based on the present information available. These recalculations may either be performed through numerical perturbation, or via algebraic differentiation of known mathematical models describing the prediction and sensor update processes. Recalculation of the jacobians may therefore increase the computation time of the Kalman filter, to an extent dependent on the complexity of these process models. This however results in a significant increase in flexibility and accuracy in the estimation of any nonlinear systems.
2.3.5 Unscented Kalman Filter (UKF)

For highly nonlinear systems, the extended Kalman filter may result in inaccurate estimates, as the algorithm models nonlinear functions through repeated linearisation around target points. The unscented Kalman filter (UKF) [18] reduces this inaccuracy by sampling and propagating a distribution of test points through the prediction and/or update models. This process therefore involves a wider consideration of local nonlinearities in the models than the EKF, improving tolerance to nonlinearities.

The downside to the UKF formulation is a significant increase in computational expense. As this system relies on propagating many test points to generate sensitivities, this method is similar to the calculation of jacobians via numerical means. This is much more computationally demanding than using algebraic differentiation methods. Furthermore, the formulation involves a minimum of two test points for each parameter in the state vector, resulting in a heavy computational penalty for large numbers of states.

2.3.6 Particle Filter

One of the fundamental assumptions of the Kalman filter is that all noise is gaussian, and with zero mean. If this is not the case, a Kalman filter will give poor performance. The particle filter is an extremely computationally demanding solution to this problem, which involves generating a large number of filter seeds distributed randomly around the present estimate. These seeded solutions are then propagated forward, before being compared with new sensor measurements. This comparison can then be used to rank the accuracy of the original seeds. At this point, poorly matching seeds may be removed and new seeds created, which better fit the sensor measurements before re-propagation. As this process employs massive redundancy, in order to model nonlinear and non-gaussian systems, it requires a large amount of processing compared to the more standard Kalman filters. Furthermore, the number of seed points required will be highly dependent on the number of parameter states, increasing computational requirements rapidly with increasing number of states.
2.3.7 Information Filter

The Kalman filter models uncertainty based on the covariance matrix. This matrix describes how well parameters are known by assigning each a variance, i.e. standard deviation squared. The parameters are also assigned cross-coupling terms, which define how each term relies on each of the others. For this reason, a Kalman filter can struggle to model any parameter which is completely unknown, or otherwise would have infinite variance. This is because computer systems tend to struggle when managing undefined quantities, such as infinity. The information filter [19–21] attempts to solve this issue by modelling the inverse of uncertainty, or the amount of information contained in an estimate, via an ‘information matrix’. This therefore allows unknown parameters to be successfully modelled (they contain no information, hence zero). The drawback of this technique is that the system loses the ability to model perfectly known parameters, which would contain infinite information.

The second advantage of the information filter is that it significantly simplifies the computation of the Kalman update step, which would traditionally require a matrix inversion. The information filter update step requires no such computationally expensive operations. Instead, the drawback to this method is a more complex prediction step, involving a matrix division. This means that the prediction step of the Information filter is much more computationally expensive than the Kalman filter, whereas the opposite is true for the update step. This therefore limits the performance of the information filter in any system involving more frequent prediction steps than update steps. Such situations will be quite common in most feature aided inertial navigation systems.

2.3.8 Simultaneous Localisation and Mapping (SLAM)

The simultaneous localisation and mapping (SLAM) algorithm, pioneered by Cheeseman [9], is an extension of the Kalman filter to allow newly discovered parameters to be initialised and estimated during operation. The idea behind SLAM is that previously unknown feature states can be initialised based on the present vehicle state estimate. These new features may then be used as relative navigation aids for the vehicle, constraining inertial drift. The result of this is the simultaneous estimation of both vehicle and feature states. Furthermore, should the vehicle return to areas it has previously navigated and views formerly initialised features, inertial drift accumulated since these features were originally initialised
can be mostly eliminated through loop closure. This act of closing the loop has the added benefit of significantly improving the uncertainties of any other features viewed in the period before the loop is closed.

Previous work has shown explicitly, that SLAM benefits from the use of fewer, better landmark observations [10]. This would therefore suggest that higher order features are beneficial for SLAM, as they may provide more information to the filter per-feature, such as heading or scale. Decreasing the number of estimated feature states will also benefit computation time.

2.3.9 Terrain Assisted SLAM (TAN-SLAM)

The SLAM algorithm can be further improved by the addition of direct fusion of known information, such as the locations of known, a-priori features. This can be performed through a basic linear or non-linear Kalman update step. This is known as terrain assisted navigation SLAM, or TAN-SLAM. The addition of absolute information into the filter, which is not coupled with the vehicle state, results in large decreases in state variances for both vehicle and features. This is therefore highly beneficial to the navigation aid system, as it enables residual drift from the relative SLAM navigation aid to be removed, without the path-planning requirements of loop closure.

2.3.10 Observations

A number of observations can be made in relation to commonly used sensors and data fusion systems:

- Although aerial platforms typically feature a wide suite of sensors, these sensors individually demonstrate significant drawbacks for a number of reasons. Focus today is therefore on the fusion of these measurements, such that the shortcomings of each sensor is ameliorated by advantages of others.

- Despite the clear advantages of robust, stealthy navigation, sensors which could allow self-contained, passive localisation, such as digital cameras, are not commonly found on aerial systems. This is in part due to the relative simplicity of traditional GNSS localisation. Visual processing also generally carries a high computational expense, whereas GNSS signal processing is comparatively
straight-forward.

- Simultaneous localisation and mapping is a convenient method by which visual data can be used to help constrain inertial drift. This is currently a popular topic of research in many different navigation fields, including with aerial platforms.

- Traditional SLAM implementations assume that any operation environments are completely unknown. With the quantity of environment information freely available today, this is rarely true, making this assumption nonsensical. It is clearly beneficial to use any available data to improve the navigation solution.

- In some cases, certain types of a-priori knowledge about the environment can be used to evaluate absolute position solutions. This is highly beneficial, as SLAM can only limit drift, whereas the injection of absolute position innovations can result in the reduction of uncertainty to that of the a-priori known features. This process also has the secondary effect of improving the position estimates of recently acquired SLAM features.

- The EKF is a very common filtering algorithm, which is tolerant to a reasonable amount of nonlinearities in the modelled systems. Other more complex methods, such as the UKF or particle filter, may theoretically be more accurate and robust, however this comes at a steep cost to computation time. It appears likely that the use of an EKF will be necessary for any real-time implementation of a SLAM based visual navigation system, especially if the number of estimated feature states should become large.

### 2.4 Contour Based Aerial Navigation Methods

#### 2.4.1 Contour Based Terrain Aided Navigation (TAN)

A well-known aerial localisation algorithm which utilises a terrain aided navigation system (TANS) is TERCOM [7, 8], where active radar terrain scans are associated with an existing terrain map. This system therefore enables GNSS-free localisation for vehicles, especially cruise missile systems. A similar method, more frequently employed on manned military aircraft, is TERPROM [16]. This system can use a number of different active sensors, such as infrared, laser or radar, to determine the range to terrain and
develop a contour map. The reliance on active sensors makes these systems vulnerable in many military and combat scenarios, due to the relative ease by which active systems could be detected by hostile forces. Furthermore, active sensors are high power devices which also add substantial weight, volume and cost to a platform.

Despite the shortcomings of active TANS systems, this area has invited a large amount of research towards improving robustness. This predominantly is achieved through data fusion of TANS sensor measurements with inertial measurements, and in some cases GPS measurements if available [22, 23]. More advanced methods of fusion using a non-linear Kalman filter [24] have also been demonstrated. This involves the direct fusion of radar based terrain range measurements with inertial navigation techniques. Kalman filtering of these measurements improves robustness by tracking motion during non-level manoeuvring, where radar range measurements may not be available. Furthermore, it provides continuous position updates, as well as tracking the accuracy of the solution through covariance estimation. More recent work towards further robustness improvements involves the use of unscented Kalman filtering to improve the linearisation of the sensor models.

2.4.2 Visual Contour Based TAN

Methods of replacing the active sensors used in TERCOM-like TAN systems with passive equivalents is currently an active area of research. This would carry benefits to platform weight, size, cost and power draw, as well as increasing difficulty of vehicle detection by hostile forces. Such a system would also potentially be more robust, due to resistance to jamming and greater sensor operational range. These benefits make such a system highly beneficial to aerial implementations, especially for military applications.

The use of stereo cameras to infer terrain profiles has been investigated for lunar landings [25]. These profiles are then matched to a known DTEM by using the distribution of detected peaks, such as hills or mounds. Data association in this work is performed by the comparison of triangles defined with vertices at different permutations of three peaks. The triangle side length is used to compare features, with position, scale and orientation used to localise the vehicle. A single pose estimate is recovered through polling of each associated triangle for the most common vehicle state solution. This method therefore requires a
high resolution DTEM, as multiple peaks must be resolved in each camera frame in order to infer a pose estimate. Furthermore, as this method relies on stereo imagery, the platform must be sufficiently large to allow a significant angular difference in terrain viewing angle between cameras.

Monocular cameras may be used in place of stereo imagery to infer range. This substitutes the constant baseline separation between cameras in the stereo setup with a time-dependent separation, resulting from the motion of the vehicle. Relative motion of camera imagery, relative to the known motion of the vehicle, is used to infer range. This motion of objects in the camera frame is determined using optical flow techniques [26], and can be used to generate terrain profiles.

Monocular camera terrain profile estimation has been demonstrated previously, and used to perform TAN. Data association between the estimated terrain profile and a known DTEM can be performed by taking a 1D terrain slice [27, 28], and finding the position where this slice best fits the DTEM. This method however, ignores much of the available information and therefore will require high accuracy and resolution terrain maps. This quality information may not be available in most areas, restricting the use of such a simple implementation.

As an alternative to using contour profile matching to provide position updates for a vehicle, a DTEM may instead be used to estimate how a vehicle has moved between camera frames [29, 30]. This can be performed by estimating where sampled points in the camera frame would lie on the DTEM. Optical flow may be used to determine how these points move between camera frames, before predicting the vehicle motion required to affect this change. The problem with this idea is that it will be highly sensitive to position error, which may cause the state estimate to diverge. Furthermore, as this method only provides relative motion estimates instead of absolute position, the vehicle pose estimate will still drift over time.

2.4.3 Horizon Based TAN

Terrain contour matching systems can have difficulty with localisation when operating at low altitudes. This is due to the small amount of underlying terrain that will be visible to the vehicle, which may not be sufficient to return a strong contour association. Low DTEM resolutions can also hamper the effectiveness of contour matching, as the point sample spacing may be insufficient to uniquely describe the terrain. A solution to both these problems can be found by instead employing the DTEM to estimate
the horizon profile, which can be used to provide localisation information [31–41]. The drawback of this method is increased complexity, due to the viewpoint dependence of the horizon profile. For example, small changes in vehicle altitude can significantly change the shape of the current horizon, especially at low altitudes, near mountainous terrain. Furthermore, it is impractical to use typical range sensors to estimate the distances to horizon features. This is because the true range to the horizon can often be in the hundreds of kilometres. Active sensors, such as radar, do not have the power to obtain readings at these distances. Alternative passive methods, such as stereo imagery or optical flow, cannot obtain sufficient angular separation for an accurate reading.

2.4.4 Observations

A number of observations can be made in relation to the state of research towards terrain profile based navigation systems:

- Radar based terrain profile estimation and matching systems have existed for decades, such as TERCOM. These systems however, exhibit significant drawbacks due to the use of a high powered active radar sensor. The most critical of these drawbacks for military applications is that the radar system acts as a beacon to any potentially hostile forces, making stealthy operations impossible. For civilian operations, the power draw, volume and weight of an appropriate radar range finder place undesirable constraints on any aerial platform.

- Passive visual camera sensors can be used to replace active radar systems by using stereo cameras. This however generally comes with a significant loss of accuracy, depending on the available angular separation of the cameras.

- Instead of using binocular cameras, the motion of the vehicle can be combined with the apparent motion of features in the camera frame (optical flow) to infer range. There has been very limited investigation into employing this method to estimate the altitude profile of terrain. The creation of a visual alternative to TERCOM would be a natural extension of previous work. This would allow the vehicle position to be estimated through data association with a known a-priori terrain contour map.
• A known terrain map can be used to estimate the motion of a vehicle through the apparent motion of the underlying terrain. This is known as visual odometry, however as its reliability is dependent on the accuracy of range estimates to terrain, operation using this method risks solution divergence. This is because position errors can cause range-to-terrain estimate errors, which in turn cause velocity estimates to be incorrect.

• A simultaneous navigation and estimation technique could be employed which uses visual odometry to constrain inertial drift, while relying on terrain profile match innovations to prevent solution divergence.

• The apparent shape of the horizon can also be used to localise an aircraft, however due to the likely distances between vehicle and horizon, accuracy of this method is likely to be low. Furthermore, changes in lighting, cloud or visibility can significantly affect the profile of the horizon, causing issues with reliability.

2.5 Visual Feature Detection

2.5.1 SIFT/SURF Methods

Common feature types used in both visual robotic and aerial navigation system research are the SIFT [42] or SURF [43]. These features share many similarities as they are both highly abstract, point features designed for ease of use in computer based visual data association. These metrics are tolerant to scale and rotation changes, as well as exhibiting partial tolerance to other affine transforms and changing lighting conditions. The use of point features such as SIFT and SURF for data association can result in problems due to their low level definitions. This is because SIFT/SURF are described by a large number of different parameters. Small amounts of noise in each of these channels can accumulate to result in large apparent errors, leading to false negative associations. Their abstract nature also leads to them having few parallels with human recognition techniques.
2.5.2 Texture Segmentation

Generally, images consist of a number of different objects, which can be distinguished based on a range of properties, such as colour. Other methods of distinguishing objects involve analysing differences in variations of colour and brightness, or pattern variations. This is known as texture segmentation, and is a very complex and problematic issue to solve in computer science. This difficulty may be counterintuitive, due to the comparative ease with which humans can identify objects and boundaries in images.

There are many different methods by which a computer system can be designed to separate different regions in an image based on texture. These can predominantly be separated into four main categories:

**Edge Based Methods**

In the case where different parts of an image can be assumed to be of a different colour or intensity, it therefore follows that the border between these image regions will exhibit a sudden change in colour or intensity properties. For this reason, if the image is analysed for changes in colour and intensity gradient magnitudes, boundaries will be visible as curves of high gradient magnitude. One drawback of this technique is a reliance on differentiation of the image to find the gradient, which causes any noise in the image to be amplified. This can cause issues when analysing the result for edge boundaries, as the true edges may be hidden by this amplified noise.

**Sobel Edge Detector** The Sobel edge detector [44] is a simple method of determining the gradient of an image, while limiting the response to noise. This is performed by using a simple filter kernel that combines a centred space differentiation kernel with a small gaussian blur.

Convolving this kernel with an image will highlight regions with high vertical gradient, whereas convolution with the transpose of this kernel will resolve horizontal gradients. Taking the root sum square of these results (the vertical and horizontal gradient responses) will show the edges in the image. Similarly, the gradient of these edges can be calculated by finding the inverse tan of the two components.
Canny Edge Detector  The Canny edge detector [45] is a moderately complex method of determining edge boundaries which aims to improve the detection of continuous edges, while rejecting edges resulting from noise. This is performed using a multi-step process which begins in a similar vein to the Sobel detector, however adds extra refinement methods. These methods ensure that the detected edges lie on the strongest part of the gradient response, and also remove edges which are consistently weak. This helps to improve the continuity of strong edges, while ignoring those which are consistently weak.

Watershed Algorithm  In the situation where discrete objects are to be detected in a frame, it is beneficial to use segmentation algorithms which exclusively resolve closed feature boundaries in an image. The watershed algorithm [46] is one such method capable of separating an image into closed regions separated by edge boundaries. This can be thought of as if the gradient magnitude was a height relief map mounted horizontally, with water raining onto the surface. Regions which collect water, or catchment basins, can be thought of as concave bounded areas that can be classified as unique segments of the map. These regions can therefore be used to separate different objects in an image, as demonstrated in [47].

Region Based Methods

Alternatives to edge based methods are region based algorithms that function a logically opposite way. Region based methods rely on the detection of areas with high homogeneity, which are then concluded to be of a single texture or object. Edges between regions are therefore assumed to not be located in homogeneous regions, but to be the borders between such regions.

Competitive Region Growing  A popular region based texture segmentation method is competitive region growing [48], where an image is analysed for points of minimum variation, ie. high homogeneity. These points are defined as seed points, which are iteratively expanded into adjoining homogeneous regions. This expansion continues until regions meet, allowing the border between these areas to be defined as the segmented edge. The main drawback of this method is that due to its iterative nature, it is comparatively computationally expensive.
Pixel Based Methods

In some cases, it is important to separate an image into different element types, without assuming similar elements will be clustered together. Examples of this would be detecting sparse trees or shrubs in a paddock or desert, or locating cars or cattle. These objects will feature similar properties, but may appear anywhere in an image, and may be disconnected.

Clustered Statistical Segmentation A method of detecting these feature types is outlined in [49], where statistical analysis is used to characterise each pixel in the image based on the surrounding pixels. The characterised pixels are then clustered into a number of bins of similar pixel properties. For each of these bins, the image location of each pixel contained within may then be determined. This therefore results in the determination of what parts of the image have similar properties, i.e. where similar objects are located.

Partitioning Methods

A fourth method of image segmentation is through analysing regions of an image, and making a decision about whether this region contains a boundary, or not. This therefore will allow the iterative division of non-homogeneous regions in an image. Division may be continued until a certain required resolution is achieved.

Co-occurrence Homogeneity Decisions on whether a region is homogeneous, or if it contains edges, can be made through the analysis of co-occurrence matrices, as in [50]. These matrices describe the relative distribution of pixel values over a desired region, and therefore can be used to describe the texture properties contained within these segments. Non-homogeneous regions may then be subdivided and re-analysed. Alternatively, wavelet transforms can be used [51] to assess regions for homogeneity before subdivision.
Specialised Methods

There are a number of other methods which can be used to segment images, which are mostly designed to specifically target a particular texture type, such as water. Two examples are discussed below:

**Terrain Feature Reflections**  In a situation where a vehicle is navigating a riverine environment using a forward facing camera, it can be assumed that the visible water will reflect objects which are on the riverbank, such as trees. These reflections can be used to estimate the water edges [52], by analysing these reflections for planes of symmetry and discontinuities. This method is valid for both low-flying aerial vehicles, or autonomous boats. This method would therefore have difficulty distinguishing the boundaries of choppy water, which would disrupt these reflections. Terrain reflections will also only be available with certain viewpoints and environments, preventing this technique from being used in a general case. Finally, the changing perspective effects of reflections will cause uncertainties in the position of the water boundary.

**Reflection Intensity Gradients**  Autonomous off-road ground vehicles generally need to detect and avoid water puddles, which may not reflect anything other than sky. The variation of water reflectivity with incidence angle can be used to identify water in these situations, as shown by [53]. Intensity gradient characteristics of water body reflections can be estimated from the incidence angle of the viewed ground. Regions exhibiting these intensity characteristics are identified, and assumed to be water. Clearly, this method is highly dependent on an oblique viewing incidence angle of any water bodies. For generalised aerial navigation, this is unlikely to be the case when operating at typical flight altitudes for even small aerial vehicles.

2.5.3 Deep Learning Methods

A promising new area of computer science involves deep learning, with the use of neural networks. These can be described as complex layers of filters, which aim to replicate the highly branched and interconnected nature of biological brains. A derived process applicable to imagery and computer vision is the convolutional neural network (CNN). This involves filtering an image using multiple broad arrays of
specifically tailored filter kernels, i.e. convolutions. These systems have been demonstrated to be highly proficient at performing a wide range of computer vision applications. For example, the identification and localisation of desired objects in imagery [54]. This work outlines a method by which specific aquatic creatures (dugongs) can be detected from an aerial vehicle, despite choppy water and adverse variations in seabed colours. An example of the detection of road intersections in imagery is presented in [55], demonstrating an excellent detection ratio of these features.

Deep learning CNN techniques can also be used to separate images into distinct regions, or highlight the borders of individual objects. Image segmentation and region labelling has been shown by [56], using an extensive set of varying images and region types. The use of CNN methods to segment image textures is also presented in [57], which provides an evaluation of deep learning techniques.

The use of existing database maps can be used to enhance the effectiveness of deep learning by readily providing extensive, verified, training data. This is demonstrated in [58], using a road detection example, while also presenting methods of improving the robustness of the neural network learning process.

The main drawback of deep learning techniques is the computational expense inherent to their implementation. Training a CNN can take hours to several days [56], furthermore the application of the trained system to imagery, can also be highly computationally expensive. High performance, massively parallel hardware (such as a modern enthusiast or professional grade GPU) is essential to provide near real-time image processing. For example, the system presented in [56] is claimed to produce on average 8 frames per second, using a very expensive NVIDIA Titan GPU, which also suffers the drawback of a high power draw. Every year, such computing devices are re-developed and updated, ensuring increased performance, efficiency and affordability; however at present, these demands limit the practicality of implementing deep learning methods.

2.5.4 Observations

A number of observations can be made in relation to the detection of visual features in camera imagery:

- The most common feature types used in SLAM algorithms are abstract features such as SIFT or SURF. These do not have parallels with human visual recognition techniques, and are unlikely to
be robust to changing lighting conditions.

- A number of different types of segmentation techniques have been discussed which separate images into visually distinct parts based on colour and texture. Of these, the edge based methods appear to be the most promising, as they specifically target the types of features which are most distinct for natural feature navigation, i.e. edges. These methods therefore result in the definition of distinct regions, which may be classified as a separate process.

- If edge features are to be determined in an image, a wide range of different algorithms and methodologies can be employed to find these edges. These different algorithms are designed to find different kinds of edges, in different imagery conditions.

- Unfortunately, some of the promising, water-specific segmentation techniques cannot be used for general aerial navigation, as detection of edge boundaries in this case is mostly occurring at approximately normal incident angles. Therefore, any sky reflections are likely to be uniform, apart from clouds and solar reflections, and no terrain is likely to be reflected. Furthermore, as these methods are exclusively designed for detecting water, they are incompatible with navigation over other visually distinct edges, such as roads, or forest boundaries.

- Most edges in nature describe closed features, such as lakes or forest boundaries.

- The watershed algorithm is a convenient method of determining edges in an image, which inherently focusses on closed edge boundaries.

- It is beneficial to determine what type of ground texture exists in each part of an image. This information can be used to help categorise any detected boundaries, which will generally be located between regions of differing texture. Classification techniques should therefore be used to ascertain the texture types of different regions.

- Deep learning methods, such as the CNN, are a promising proposition for the future of image segmentation. However, computational demands limit their practicality for real-time implementations at present.
2.6 Feature Based Aerial Navigation Techniques

Methods capable of providing GNSS independent localisation for unmanned aerial vehicles (UAVs), have existed since the 1950’s, with the creation of the terrain contour matching (TERCOM) [7, 8] system. Radar generated terrain profile maps are compared to prior known height contour maps, allowing vehicle localisation. This reliance on radar leaves the system completely dependent on heavy, expensive and power hungry equipment. Furthermore, the system can be easily detected by potentially hostile forces during operation. Research focus today is therefore towards the use of passive, self-contained systems, such as computer vision algorithms, to localise these vehicles. More specifically, research is towards the use of discrete features for localisation, such as objects, rather than continuous data sources, such as terrain profiles. Navigation using discrete features is beneficial as they have the potential to minimise computational complexity, while providing more definitive localisation information. This is because individual features are likely to require less data for classification, and may be more precisely located.

2.6.1 Point Feature Methods

The SLAM algorithm (introduced in section 2.3.8) allows relative navigation using previously unknown features, and therefore limits the accumulation of integration error, when features are visible. Features are characterised, estimated, and their motion relative to the vehicle used in vehicle localisation. The use of SLAM to help in both ground robot and aerial vehicle navigation is an active area of research interest [1, 52, 59–98] in the research community. This is largely due to the lack of reliance on any a-priori information, such that SLAM-enabled vehicles can operate in completely unknown environments. The prevailing focus of recent research and development towards SLAM-based aerial navigation aids use simple, low level features, such as point features [99]. This limits the amount of information that can be obtained from them, reducing the performance of the navigation aid.

Visual SLAM implementations result in extra challenges over non-visual radar based methods, due to the lack of the intrinsic range information which comes from the use of a radar based system. This bearing only SLAM therefore requires special handling of any new features, as the absence of any range information is difficult to model in the estimation and feature initialisation process. Feature parametrisation techniques can be used to achieve tolerance to the lack of range information [66, 68, 71,
79, 82, 88, 93]. Other methods involve delaying the initialisation of new features until range information can be inferred through triangulation from vehicle motion [76, 82].

**Abstract Features**

Aerial navigation research using SLAM generally uses artificial markers such as traffic cones or white paper [63] deployed in-field before any aerial operations are commenced. This clearly poses practicality problems for any system which would require these features. Other features used for aerial SLAM are the speeded-up robust feature (SURF) algorithm [43], which provides a repeatable, recognisable characterisation technique. A SURF however, is a highly abstract feature, and has no parallels with human feature recognition. It is also not temporally robust, as changing lighting or viewpoint conditions may cause the characteristics of the feature to change significantly, rendering it useless as a navigation marker. Also, SURF are a collection of many different descriptive parameters, and as such the cumulative effect of noise on each parameter can have a detrimental impact on robustness. Finally SURF are point features, and are therefore not useful for characterising most natural features, which are better described by boundary curves, such as lake and river edges, or forest boundaries.

Other examples of easily recognisable robust visual features which could be used for SLAM navigation are building roofs of specific colour characteristics. However, as these generally only exist in urban areas, this places undesirable restrictions on missions. Furthermore, as these would be in most cases described as point features, and may not be unique, problems with data association and heading drift will degrade system performance.

**Riverine Mapping**

Using SLAM to map riverine systems has been demonstrated previously [52, 61] using a MAV helicopter and a small boat respectively. These systems however make a number of assumptions which hamper their use for general visual navigation. Firstly, an assumption is made that the vehicles will be operating on (or flying low over) a river, with a forward facing camera. Therefore the assumption is that the centre base of the camera frame will contain the river as a planar feature, therefore helping texture segmentation routines determine the river edge. Secondly, these systems still use abstract SURF associations for SLAM,
whereas the river edges are simply mapped, and not used for localisation.

2.6.2 Edge Feature Methods

SLAM using non-abstract edge features has been investigated using BS-SLAM [85, 91, 92], which uses the basic spline (or b-spline) to navigate in GNSS denied environments. This uses a laser range finder to scan distances to desks and other barriers in office environments, with a spline model used to define these features. The work however is limited to two dimensions, and the use of a laser range finder poses practicality problems for an airborne system. These problems are due to the weight, cost and power draw of such a system capable of operational ranges of over 1 km, which are likely for aerial navigation altitudes. Furthermore, the reliance on range information neglects many of the clearest human aerial visual navigation features, such as rivers and roads. Such a system would also have difficulty operating over flat terrain due to the lack of any contour definition which could be used for data association and localisation.

2.6.3 Map Aided Navigation

As the SLAM process relies on building a relative map to limit inertial drift, the vehicle pose solution will still drift over time due to uncertainties and noise in any sensor measurements. Instead of relative measurements, the fusion of absolute data can be employed, such as the detection of features with known positions. This information allows the filter to eliminate inertial drift, as opposed to temporarily halting the accumulation of drift as with SLAM. Fusion with absolute data is therefore vital for successful navigation, as operation without this information will inevitably lead to a continuous decrease in solution accuracy.

Imagery Based Map

Geo-referenced satellite or aerial imagery provides a convenient database of absolute position information for localising an aerial vehicle [27, 100–104]. High resolution imagery is freely available for most of the Earth’s landmass via services such as Google Earth [105] and Google Maps [106]. Data association
with features detected in these databases can therefore be used to estimate the current pose of the vehicle. Drawbacks of this method are generally related to robustness, as changes in lighting conditions or physically altered terrain can cause rejection of correct data associations, or even false positive matches.

A method of association between an aerial imagery database and on-platform camera images is shown in [27, 103], as the basis for a visual-only navigation system. Data association is performed through aligning edge points based on image registration. This imagery based localisation is also combined with stereo based terrain profile estimation, and contour matching. Finally, the stereo imagery is also used for relative navigation through the tracking of viewed features. These three techniques are fused through a filter to optimise available data, and do not consider any inertial information. Fusion of inertial measurements would be an obvious method by which to improve the navigation solution, without any significant drawbacks. The fusion of image registration based TAN with inertial measurements is investigated in [100, 101] to help provide tolerance to loss of GNSS signals.

The direct use of aerial imagery for visual localisation is demonstrated in [100], involving template matching and image registration of stored digital aerial imagery. The drawback of this technique is that containing large amounts of high resolution aerial imagery requires a large amount of data storage volume, and furthermore template matching is a comparatively computationally expensive process. Therefore this system is likely to be quite slow and require a large amount of extra computing hardware, especially when operating over wide areas and long distance missions.

**Feature Based Map**

The direct use of aerial imagery for visual localisation can be problematic due to computational complexity of template matching. It is also limited by robustness issues caused by matching schemes often based on aligning binary edge pixels, which are very low level features containing minimal unique information. These issues can be resolved through the use of databases consisting of discrete features. An example of this is outlined in [102] where vision assisted inertial fusion is used to support a spacecraft landing system. The map database consists of scale-invariant feature transform (SIFT) abstract point features, which are used to localise the vehicle, and assist in constraining the motion through tracking.

Absolute position information may also be obtained through the use of a geographic information
system (GIS) as the map database [60, 107–112]. This avoids the lighting and viewpoint robustness problems caused by reliance on satellite imagery. Examples of features that can be obtained using these services [60, 107, 108, 112] are roads and road intersections, rivers, forests, buildings, etc. These features include representations of point features, edges and regions. Different features are extracted using different techniques specifically tailored to each feature type [108] before being associated with the known database, allowing vehicle localisation.

Navigation using road intersection features specifically has been demonstrated [60, 109–111]. These features have many highly desirable qualities for navigation, including relative permanence (locations do not change with time). Furthermore, road network maps are freely available, in many cases intersections are unique, and they can also often provide orientation information.

Visually distinct features may also be used to perform localisation of an aerial vehicle, if an accurate map of these features is known beforehand. This is known as feature-based terrain aided navigation (TAN), and has been previously demonstrated to have the ability to accurately localise aerial vehicles using visual information. Previous work includes navigation using road bridges [59], using a rule-based data association method to recognise features and localise the vehicle. Other work includes the use of road intersections [60]. These junction features are detected using a maximum likelihood classifier based on their spatial geometry. The drawbacks of these two methods involve the choice of features, which mostly exist in urban environments, and may be too sparse for successful navigation in regional areas. This limits the areas where this system could be deployed, hampering usefulness.

2.6.4 Terrain Aided SLAM Fusion

The ideology behind SLAM is to collect all information about an environment during system operation, from the initial condition that the environment is unknown. As demonstrated in section 2.6.3, it is common to have a-priori information about an environment that can be used to localise a vehicle. It therefore becomes both possible and beneficial to mix the concept of SLAM with map-based localisation techniques. This allows the system to maximise the use of available information. During periods without visually distinct features, pose estimation can rely on inertial information. Should previously unknown features be seen, SLAM can be employed to limit the accumulation of inertial drift. Finally, should known
features be identified by the vehicle, this absolute position information can be used to fully localise the pose estimate.

The idea of fusing SLAM based navigation with map-based localisation techniques is not commonly used, despite its obvious benefits. The road intersection work proposed in [60] uses this idea in an aerial application, however other work in this area is generally for ground based robotic systems [86, 90, 94, 96, 97].

2.6.5 Observations

A number of observations can be made in relation to the state of research towards visual feature based navigation systems:

- The majority of work involving airborne SLAM implementations rely on artificial markers manually placed in the environment, or abstract point features such as SURF. This is in direct contrast to features a human pilot would use to successfully navigate an aerial vehicle. More work towards using higher level natural geographic features would bring these systems more in-line with human navigation techniques.

- As SLAM operates via the creation of a relative map, the ability of this algorithm to limit inertial drift is impacted. The effects of the residual drift in the SLAM process can be minimised through repeated loop closure, or by limiting operation time. Both of these solutions may place unacceptable constraints on any mission the platform can undertake.

- If SLAM is to be successfully used for long-term navigation, the ability to fuse innovations relative to a-priori map database information will be required. This allows the pose of the vehicle to be observed directly, placing a bound on position error and uncertainty. The use of known data is therefore critical to any implementation of airborne SLAM.

- Aerial and satellite imagery is commonly used to provide absolute position data through template matching. This is despite this process being problematic due to changing lighting and other operating conditions affecting the robustness of such a system.
• The use of higher level features for localisation has only received minor investigation, despite benefits to robustness. Furthermore, data association between discrete feature databases will result in computational expense benefits over the continuous data exhibited by raw aerial imagery. What work there is towards using higher level features tends to rely on road intersections, which may result in problems when navigating in rural areas where road intersections are less common.

• A more general visual navigation system which operates through edge curve based SLAM would present itself as a natural progression of previous work. Such a system would be capable of navigating based on a range of visually distinct features such as rivers, lakes, roads, forest boundaries, etc. It would also require the ability to fuse absolute position innovations through associations with a known geographic feature database. Finally, inertial information would be used to estimate vehicle motion in any areas without visually distinct features, maximising the use of available data and sensor measurements.

2.7 Data Association

Data association methods vary greatly and are dependent on the type of feature which needs to be associated. Furthermore, for curve based association the methods may vary based on how the shapes have been modelled. Many diverse methods of curve modelling exist to simplify the data required to describe raw data, best described as a linked pixel list. These methods include polygon and polynomial approximations, as well as frequency methods such as Fourier or Taylor series decomposition.

2.7.1 Spline Node Matching

A further subset of methods which display a number of clear beneficial properties are splines. Of these, the cubic basic spline (b-spline) is popular as it is simple to implement. A small set of coefficient values at control points are determined, which contain the shape information of the original curve. This benefits from being a purely algebraic curve descriptor, thereby allowing curve derivatives to be accurately calculated with ease. The extraction of derivative information is vital, as this contains the position-independent shape information of the curve, which is used for association. It also allows extra coefficients to be used to increase information density where it is needed to describe complex curve
shapes, while minimally describing simpler parts. Also, the b-spline features a high degree of local controllability, i.e. altering coefficient values will change the shape of the local curve, however will not affect any areas not adjacent to this coefficient. This therefore improves the robustness of the spline. These benefits make the basic spline a clear area of interest, especially in researching robust matching techniques.

Matching b-splines can be performed by directly comparing coefficient values, however this is difficult as spline coefficient values are non-unique. As a particular curve can be represented in a large number of different coefficient spacings and values, identical shaped splines may have vastly different coefficient representations. A method of re-estimating coefficient spacing and values to be more robust is proposed by [113], in order to improve the compatibility of splines for association. Curves can then be compared using weighted b-spline moments, allowing association and the determination of any affine transform between matched curves. Applying this transform to one of the curves allows the coefficient values of the splines to be directly compared. Benefits of this method are a tolerance to affine transformations, partial occlusions and local noise. This matching algorithm is, however, let down by intolerance to inclusions or other local dissimilar regions. Furthermore, this method is still reliant on direct coefficient matching, and as such small changes in the spline shape may cause large inconsistencies between control point coefficients.

2.7.2 Spline Curvature Matching

Association methods which focus on the algebraic and continuous nature of the spline derivation are preferable to direct comparison of control point coefficient values due to robustness. Curvature and curvature change are powerful comparison metrics, as they are independent of rotational orientation, and if correctly normalised, also unaffected by scale differences. Most importantly, as curvature is based on distance along the curve, it is independent of control point spacing. Furthermore, differentiation of the algebraic formulation of splines results in a simple algebraic expression for calculating the curvature of splines at any point. Recent advances in curve matching have therefore focussed on curvature based association, and the determination of affine transforms between matched features.

Curvature scale space (CSS) [114] is one method which relies on curvature based comparisons
between shapes. Curves are repeatedly filtered with increasing filter kernel sizes, with inflection point locations recorded at each point. As the curve is filtered, these inflection points move, and ultimately combine and cancel when specific features are filtered out. The path these inflection points take, and the filter coefficients required to delete them, develop a two-dimensional curvature scale space trace. This becomes a simple, normalised descriptor of the curve shape which is tolerant to rotation and scale differences, as well as reasonable levels of noise. This trace can therefore be used to compare curves and develop both associations, as well as affine transforms between the two features. The shortcomings of this method are predominantly due to the trace being normalised over the known length of the curve, and as such this method is designed for matching closed features. Furthermore, as any occlusions or inclusions in the feature will alter this length, this method is intolerant of such errors. Finally, the repeated filtering of the curves result in a heavy computational expense to the method.

Association of open curves can be performed through characterising the shape using a curvature based signature [115]. This defines a new shape based on the curvature, and the integral of the curvature magnitude, creating a characteristic signature which is independent of translation, orientation and scale. This signature can then be compared to similar signatures calculated from other match candidate curves. A minimum cost match position can be determined through the use of a sliding window technique, determining the offset which results in the highest similarity between signatures. This method is tolerant to matching incomplete curves, however the method is not tolerant to either occlusions or inclusions, as these will greatly alter this signature.

Tolerance to partial occlusions or inclusions can be obtained through characterising parts of a curve separately, instead of the entire shape as one feature. This is demonstrated using invariant curve descriptors [116] on segments of a shape. These curve descriptor signatures can therefore be matched to signatures from a curve database, thereby allowing successful data association. This method also allows for the determination of the affine transformation between curves.
2.7.3 Observations

A number of observations can be made in relation to the state of research towards the association of edge based features:

- Due to their algebraic formulation, data-compact and continuous nature, the cubic basic spline is a common and useful method of characterising arbitrary curves.

- The disadvantage of the b-spline is that node placement is non-unique. This causes problems with data association, as similar curve shapes may be described by vastly differing node coefficient values.

- Methods exist which attempt to standardise node placement, in order to improve ease of matching based purely on coefficient values. These methods however, are still fragile to noise in some conditions.

- More robust spline matching techniques rely on curvature of a spline, as this provides a matching scheme tolerant to affine transforms and bypasses the non-uniqueness of the coefficient values.

- Tolerance to occlusions, inclusions and branches can be provided by separating the spline into segments based on some specified criteria. Data association may then be performed on these segments individually. This prevents specific areas of curvature difference from affecting associations between areas which are of the same shape.

2.8 Summary

The main purpose of GNSS in the case of aerial navigation is to provide absolute position information. This allows inertial integration drift to be negated, while still providing accurate, continuous solution estimates of the vehicle pose. The result of this is that the navigation system is completely reliant on GNSS, and is therefore not robust to GNSS unavailability. The main alternative to GNSS position updates is TERCOM, however reliance on active radar results in a number of undesirable restrictions on both possible mission operations, and aerial platform specifications. It would therefore be highly beneficial to introduce a new system which is self-contained and also is tolerant to external conditions.
Furthermore, a system that operated exclusively using passive sensors would also be greatly beneficial for stealth-capable vehicles, while simultaneously potentially limiting the power draw of the system.

A potential method of achieving these goals is to employ digital video cameras in order to collect data about the environment, assisting in the navigation of the vehicle. Visual camera systems have the benefit of being light weight, exhibit low power draw and are relatively inexpensive, as well as being passive sensors. Should processing be performed on-board the vehicle, this would also provide tolerance to electromagnetic interference.

Visual navigation can be performed using the SLAM algorithm, which does not require any information about the environment to operate. Previous implementations of aerial SLAM function using abstract or artificial point features. This risks robustness problems stemming from changing lighting conditions, or from using low-level features which may not be unique. It would be far more beneficial to operate a SLAM filter which tracks higher level features similar to those that a human pilot would use. Edge curves are a prime example of such features, as they are likely to contain unique information which can improve the robustness of any data association. Furthermore, the choice of edges representing physical boundaries in the terrain such as river edges and roads will increase tolerance of the system to changing light conditions and viewing angles.

Although the use of SLAM is a benefit to navigation systems as it limits inertial drift, the drift itself is still unbounded with time. This means that over protracted missions, a SLAM-only localisation solution will degrade to the point of uselessness, albeit at a much slower rate than without SLAM. In order to properly bound position drift or negate it entirely, absolute position innovations are needed. As an edge-feature-based visual SLAM system will be constantly initialising and estimating the shape of viewed boundary features in the environment, it is logical to use these for non-relative position updates as well. This can be performed by assembling an a-priori map of edge features using a secondary source, such as geo-referenced aerial imagery, with which to associate viewed features. This method would exhibit benefits over the use of aerial imagery directly, due to improved data storage requirements, and because of the use of discrete, rather than continuous data association algorithms.

Data association between curve features can successfully be performed through curvature based methods, using basic spline models. Reliance on curvature, however, increases the effects of any
sensor noise in the feature shape estimates, as differentiation tends to amplify noise. It may therefore be beneficial to rely on matching based on curve heading angles, as well as changes in heading. Robustness of association can also be improved through the consideration of small segments of an edge feature separately in the matching process. Otherwise, occlusions or inclusions can cause false negative associations, even if the majority of two curve features correlate well. Further aids to data association can be obtained through the consideration of the ground texture type on either side of the border features. This will assist in decreasing the occurrence of false positives, as it will prevent associations between features of different types. An example would be rejecting a possible match between a water/grass boundary and a grass/road boundary. Should a road be located on the shore of a lake, these two edges will likely have the same shape, however there will exist a position bias between them. Mistaking one for the other would therefore potentially cause the solution to diverge, and must be avoided.

The use of edge features to navigate an aerial vehicle will lead to a sparsity of observation information when operating over terrain which does not exhibit such features. It would therefore be advantageous to also use the profile shape of underlying terrain to obtain absolute position information. This could be achieved through the development of a strictly visual implementation of the principles behind the TERCOM system. As optical flow can be combined with knowledge of the motion of a vehicle to ascertain range to viewed objects, optical flow can be used to estimate the shape of terrain, without any active measurement devices. This profile estimate can therefore provide absolute position estimates through data association with a known DTEM.

Optical flow can also be used to provide visual odometry measurements, assisting the visual navigation system in tracking the vehicle velocity. This can be done by considering the expected range to terrain based on a known terrain height map, as well as the estimated position and altitude of the vehicle. As the optical flow measurement returns the angular rate of features moving past the vehicle, this information can be used to determine the true velocity of the vehicle, without the effects of wind. Visual odometry can occur simultaneously with terrain profile estimation, providing another method by which inertial drift can be limited between absolute position fixes.

Throughout the entire scope of this visual navigation system, robustness is a key issue. It is critically important that all efforts are taken to improve the robustness of all data association steps. Fault detection is a critical part of ensuring robustness, to help ensure that false positive data associations are not fused.
into the state estimate. Instead, as inertial data is to be used for the prediction of future vehicle states, it is beneficial to err on the conservative side of data fusion. In the case of any uncertain data association, rejection and subsequent reliance on inertial navigation is beneficial to fusion of suspicious data, which risks diverging the estimate.

Robustness can also be obtained through the separation of multiple levels of data fusion. As various types of data may be available to the system at different times, it follows that all the information available at any specific time should be fused. In the situation where no absolute position information is available, the system should still be capable of relying on visual SLAM. Also, should no visually distinct features be obtainable, it is important that the filter is still capable of operation based on inertial information. This flexibility ensures the navigation solution is as accurate as practicality allows, at all times. Furthermore, the sources of information fused into the system should be dependent on integrity, as well as availability. High reliability information fusion should be prioritised, whereas less reliable data should be ignored in the presence of any higher-confidence innovations.
Chapter 3

Background Theory

This chapter outlines some of the background theory pertaining to the development of a high level visual navigation system. This requires some related terms to be defined prior to the theory, in order to properly outline the scope of this work. Navigation is the process of determining how a vehicle must move in order to arrive at a target destination. This therefore requires knowledge of where the vehicle is at any moment in time. Tracking can be employed to determine the vehicle position based on past knowledge of its location, and how it has moved in the mean time. Alternatively, extra information obtained from the environment can be used to localise the vehicle, resulting in relative pose information. Should the vehicle state be known, path planning is the act of determining how the vehicle should transition between its current location and the target destination. The actual mechanics of following this path requires knowledge of the vehicle’s properties, and is known as guidance. Control is the manipulation of the response of the vehicle, i.e. how to make the vehicle follow the designated path.

The localisation and tracking of a vehicle is therefore a vital part of guidance and navigation, as the most critical information required to reach a desired location is the current pose of the vehicle.

3.1 Reference Systems and Coordinate Transformations

This section outlines the co-ordinate axis systems utilised in this thesis. These systems are used to describe positions relevant to aerial applications relative to differing datum points, and with various alignments.
These systems are predominantly right-handed orthogonal systems, however in some applications the use of non-orthogonal frames may be more appropriate.

3.1.1 Aerial Reference Systems

Five primary co-ordinate axis systems used for aerial navigation are outlined here. These systems relate the vehicle to a position and/or orientation relative to the Earth.

**Earth Centred, Earth Fixed (ECEF)** With an origin at the Earth’s centre, the co-ordinate axes rotate with the Earth. The $x$-axis acts through the intersection of the prime meridian and the equator. The $z$-axis points towards the north pole, and the $y$-axis acts to define an orthogonal right-handed system.

**Latitude, Longitude, Altitude (LLA)** A non-orthogonal reference frame, LLA is instead a geodetic frame which represents points on the Earth through making a spherical co-ordinate system approximation. A position on the spheroidal Earth is determined by latitude and longitude co-ordinates, with altitude defining the distance above a local height datum, such as ellipsoidal sea level.

**Earth LVLH, Navigation Frame** An orthogonal frame with the origin located at an arbitrary point on the Earth’s surface, such as at the ellipsoidal sea level height, near to where a vehicle is operating. The axes are aligned such that the $z$-axis acts towards the centre of the Earth, with $x$ and $y$ tangential to the surface of the Earth. The $y$-axis points east, parallel to the local lines of constant latitude. The $x$-axis is therefore orientated north to define an orthogonal right handed axis set.

**Vehicle LVLH** Similar to the navigation frame, the Vehicle LVLH frame is instead centred on the vehicle, with the origin located at the centre of gravity of the vehicle. The axes themselves are oriented in the same manner, with $z$-axis acting towards the centre of the Earth, $y$-axis pointing east, and $x$-axis angled north to complete the orthogonal right handed set.

**Body Frame** The body frame is an orthogonal, right handed frame of reference, with origin at the centre of gravity of the vehicle. It is distinct from the Vehicle LVLH as the axes are aligned with the
vehicle body. The \( x \)-axis points forward towards the nose, the \( y \)-axis out to the right, and the \( z \)-axis downwards out from the underside of the vehicle.

### 3.1.2 Visual Reference Systems

Two further axis systems are also used in this thesis, related to computer vision. These axis systems have been modified from standard definitions to be more intuitive, and more inline with aerial navigation axes.

**Camera Space Frame**  Similar to the body frame, the camera space frame is an orthogonal right handed co-ordinate system aligned with the camera. The origin is located at the centre of the camera, with the \( x \)-axis pointing outwards in the direction the camera is facing. The \( y \)-axis points out to the right of the camera, and \( z \)-axis out from the bottom. Figure 3.1 demonstrates these axes, as \( C_x, C_y \) and \( C_z \).

**Image Plane**  A non-orthogonal reference frame which uses a spherical co-ordinate system, outlined in figure 3.1. Position in the image frame is given by bearing \( \chi \), and inclination \( \lambda \). The \( \chi \)-axis acts to the right of the camera bore-sight, whereas the \( \lambda \)-axis is positive for positions above the bore-sight. The range \( R \) is the magnitude of the distance between the camera and the viewed point, at the \([\chi, \lambda]\) image plane position.

![Figure 3.1: Description of the two visual reference systems. \( C_x, C_y \) and \( C_z \) outline an orthogonal, camera-aligned axis. \( R, \chi \) and \( \lambda \) represent a spherical reference frame.](image-url)
3.1.3 Axis Rotation Transforms

Co-ordinates in an axis system describe a vector between the origin and a target point. In order to describe common co-ordinates in a different axis system, some measure of translation and/or rotation is essential. If the axes of the two frames are not aligned, a rotation will be necessary to describe the relative angular position. Furthermore, if the axes are aligned but the frames do not share a common origin, a translation between these different origins will be required. As this translation can only occur with aligned axes, frame rotation is vital.

Axis rotation transforms are performed using rotation matrices, which are square matrices of size equal to the number of spatial dimensions involved, e.g. three for most generalised aerial navigation cases. These rotation matrices are known as Direction Cosine Matrices (DCMs). A general DCM is comprised of three single-rotation DCMs, each one defining a rotation around a different axis. These simplified DCMs are shown below for rotations around an $x$, $y$ and $z$-axis respectively.

\[
C_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta_x & \sin \theta_x \\ 0 & -\sin \theta_x & \cos \theta_x \end{bmatrix} \quad (3.1)
\]

\[
C_y(\theta_y) = \begin{bmatrix} \cos \theta_y & 0 & -\sin \theta_y \\ 0 & 1 & 0 \\ \sin \theta_y & 0 & \cos \theta_y \end{bmatrix} \quad (3.2)
\]

\[
C_z(\theta_z) = \begin{bmatrix} \cos \theta_z & \sin \theta_z & 0 \\ -\sin \theta_z & \cos \theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.3)
\]

where $C_x$ is a rotation of the angle $\theta_x$ about the $x$-axis, $C_y$ a rotation of $\theta_y$ about the $y$-axis, and $C_z$ a rotation of $\theta_z$ about the $z$-axis.

These allow the general direction cosine matrix to be defined:

\[
\bar{x}_b = C_x(\theta_x)C_y(\theta_y)C_z(\theta_z)L_a = C_{ba}\bar{x}_a \quad (3.4)
\]
where $\vec{x}_a$ is a vector in reference frame $a$, and $\vec{x}_b$ is the vector description of $\vec{x}_a$ in reference frame $b$. $C_{ba}$ is the DCM describing the axis rotation from frame $a$ to $b$.

The reverse axis transform may be performed by considering the transpose of the DCM:

$$C_{ab} = C_z(-\theta_z)C_y(-\theta_y)C_x(-\theta_x) = C_{ba}^{-1} = C_{ba}^T$$

$$\vec{x}_a = C_{ab}\vec{x}_b$$

This allows the one DCM to convert vectors back and forth between two reference frames. It is important to note that these basic DCMs can be used in any combination, as required by the relationship between origin and destination frames of reference, and the parameters that define their relationship. For example, the DCM $C_{ca}$ can be derived from the individual transformations $C_{ba}$ and $C_{cb}$:

$$C_{ca} = C_{cb}C_{ba}$$

ensuring consideration of the order of transformations, as DCMs are non-communicative.

### 3.2 Image Processing

In order to perform SLAM, detectable, repeatable and robust features must be chosen for identification and tracking. In the case of visual spline SLAM, this requires edge features resolved using image processing. For example, in order to track splines defining the edges of water bodies, areas of water in an image must be determined. These areas are separated from areas which are determined to be other texture types, such as grass, forest or sand. Finally, the borders between regions of water and other types can be assumed to be lake or river edges. Once an edge feature has been detected in an image, it can be tracked from frame to frame, and with consideration of an estimated range to the feature, used to determine the velocity of the vehicle.
3.2.1 Texture Segmentation

Texture segmentation is the process of separating an image into segments, based on differences in their texture. It is necessary to determine what type of texture any particular viewed region is, for example areas of water, forest or grassland. It is generally a very complex step, as although the visual processing of the human brain finds differentiating regions of water from trees trivial, this is not the case for computer systems. The task then is to break down how the human brain is able to associate different arbitrary textures in an image to their physical representations, and apply this to silicon computation [117]. An alternative to this approach is to assemble an algorithm that performs this function using different methods which are more compatible with, and better suited to, general computer systems.

3.2.2 Watershedding Algorithm

The watershedding algorithm is a convenient method of segmenting an image into homogeneous regions. This is performed by operating over the intensity gradient magnitude of an image, designating regions of concavity, or ‘catchment basins’, as it is analogous to the collection of water droplets falling on a terrain surface. The algorithm is based on the assumption that areas of high homogeneity are also areas of low gradient, and that boundaries between different homogeneous regions will be of higher gradient.

3.2.3 Maximum Likelihood Classification

Segmented regions in a new image may be classified with a texture type by comparing the properties of each region to those defined by training data. This results in the separation of the image into areas of differing terrain elements, the borders of which are used to define identifiable terrain features. These areas must therefore be reduced to a predefined set of properties that uniquely describe each different texture, such as colour or frequency information. Using this information, any new image region may be analysed, and compared to these a-priori texture properties. The probability that this new region belongs to each known class may be calculated, with the maximum likelihood candidate used to define this region a texture type.
3.2.4 Training Data

In order to determine which parts of an image are different texture types, a representative sample of various texture types must be assembled with which to compare and designate texture types. This can be performed by manually designating areas of separate texture types in images similar to those that will be taken during flight operations. These samples may then be classified into a simplified set of parameters which outline the properties of each texture.

3.2.5 Over-segmentation

A significant drawback of the watershedding algorithm is over-segmentation when used on real data. This is a phenomenon where small variations in an image such as noise can result in the creation of new catchment basins. Over-segmentation is a result of the algorithm giving no weighting to the magnitude of detected gradients in the image. This leads to the image being segmented into a very large number of small cell regions, without any noticeable boundaries between these sections. This effect can be reduced through a number of different techniques:

Filtering

As over-segmentation is generally caused by noise, noise reduction techniques such as filtering will logically reduce over-segmentation. Filtering can be performed through the convolution of the image with a filter kernel such as a Gaussian, or by using a median filter, or a number of other techniques. Care must be taken however, to avoid over-filtering the image, as this will reduce the prominence of edges, making them more difficult to detect. Any 2D filtering of an image also results in a low-pass filtering of edges, changing their shape. For these reasons, filtering cannot be solely relied upon to provide a clear watershed segmentation.

A similar method involves setting a threshold on the gradient magnitude of the image, where any values below this threshold are set to zero. This tends to eliminate the effect of noise without affecting the gradient magnitudes defining the true edge boundaries. Should the gradient magnitude of the edges not be significantly larger than that of the noise, this method will result in incorrect boundary definitions,
or even complete failure to properly segment certain sections.

**Adjacent Region Similarity:**

Further reductions in over-segmentation can be performed after watershedding by considering similarities in the properties of adjacent cells. Due to particularly high noise regions, or textures containing large amounts of variation, over-segmentation is likely still present after gradient thresholding. Adjacent cells with similar colour and texture properties may be merged, as they are likely to be the same region.

**Texture Type Merge:**

As the ultimate goal of this texture segmentation process is to identify regions of an image that are a particular human-recognisable ground type, this information can be used to apply a final merging step. Once remaining regions have been designated a particular texture type through classification, adjacent regions of identical type may be merged.

### 3.2.6 Feature Extraction

Once the image has been segmented, the boundary pixels between different texture types can be extracted as edge lists. These pixel lists may then be filtered to remove high frequency noise, after which the filtered edge may be resampled at a lower frequency. This resampling aims to capture the shape information of the curve, using significantly less data than a raw pixel list. These border points may then be used as sensor measurements. Furthermore, the local in-camera boundary curve slope, and the texture types of the regions defining the border, may also be used as extra information to better describe the edge point.

The point features obtained by this curve feature extraction step form the basis of the presented spline SLAM implementation. These measurements can be used to initialise new spline features, or aid in tracking their relative motion.
3.3 Spline Curve Modelling

Curve modelling involves the definition of the metric used by which curve features are described. The methods used can help to reduce the available data to a more specific, descriptive set of information, which defines the shape of the feature. Predominantly, this reduction of information will involve some aspect of filtering, and therefore it is important to determine the length-scale that contains the useful shape information. This is also greatly beneficial for robustness and accuracy of data association of curve features.

A number of different methods of representing curve features exist, however the work presented in this thesis relies exclusively on the use of the pixel list and the b-spline. A pixel list is an extremely simple, low level method of defining the shape of a curve. It consists of vectors defining the co-ordinates of consecutive pixels along a curve viewed in an image. This method is therefore comparatively data verbose, as it is a direct port of all the edge pixels obtained from an image. This makes it a poor candidate for efficient curve modelling, and therefore should be used to regress properties of a more appropriate modelling technique.

In contrast, the spline provides a higher-level curve modelling method, allowing shape data to be reduced to a much smaller set of information. The essential feature shape characteristics can be captured by determining a desired coefficient spacing, while simultaneously reducing high frequency noise. This spacing must be chosen in order to capture the maximum desired spatial frequencies, which give these features their key human-recognisable intrinsic shapes. It is important to note that these spatial frequencies will be dependent on the range to these features, which is especially critical for the case of association. For example, at lower altitudes, high frequency information will provide more information from a smaller viewed area, and provide more precise associations. In contrast, when operating at high altitudes, lower frequency spatial information is more pertinent. Decreasing the volume of data lowers the complexity of associations, and limits the possibility of incorrect matches. Despite this reduction of information, the wider visible area arising from the increased altitude will help to ensure sufficient feature information for a definitive match.
3.3.1 Introduction to B–Splines

The cubic basic spline, or cubic b–spline is a very convenient, information compact method of representing curve features. It is effectively a one dimensional parametric weighting algorithm, which calculates a varying weighted average between four coefficient values, or nodes. When generalised to an \( n \)-dimensional case, a node is defined as a set of \( n \) coefficients, i.e., one coefficient for each dimension. A spline set is defined as at least four nodes, with each conjoined set of four adjacent nodes describing a small section of the spline curve. The spline curve itself is continuous and roughly follows the shape of the nodes. Due to this, splines can represent edge curves using far fewer data-points than a per-pixel representation. Furthermore, as node spacing can be varied, more nodes can be used in certain sections to help represent complex shapes, while fewer nodes can be used to limit data storage and computational complexity in simpler areas, such as those of low curvature. Finally, as splines are continuous they can be calculated algebraically, as can partial derivatives and gradients, greatly improving computational efficiency and functionality. The mathematical definitions of the cubic b-spline is outlined in section 3.3.2.

The main drawback of the b-spline is that they are non-unique. This means that there are an infinite number of different node positions (or coefficient values) which result in the same, or similar curve representations. This has the potential to cause problems when two splines are to be compared, as directly comparing coefficients may result in greatly varying answers, despite the true curve shapes being similar. The second drawback is that it is impractical to find an algebraic solution to a calculation of the closest point on the spline, to an arbitrary target point, for multi-dimensional cubic splines. This is a result of the third order polynomial nature of cubic splines, where 2+ dimensional solutions require the calculation of the roots of at least a 5th order polynomial. This is known to be generally algebraically intractable, and instead must be done numerically. It is therefore a reasonably computationally expensive operation.

3.3.2 Mathematical Spline Model

As previously outlined, a spline set is a continuous parametric curve governed by \( n + 1 \) node coefficients, defined by a weighting function between successive sets of four local consecutive nodes. These coefficients are stored as vector \( X_F \), which therefore defines the spline shape. The parameter \( s \) is the value of
the spline at some parametric position \( h \) along each 4-node spline segment. A global parametric variable \( t \) can now be defined along the entire spline sequence, where \( t \in \mathbb{R}[0, n] \).

\[
t = i + h
\]  

(3.8)

where \( i = \lfloor t \rfloor = \text{floor}(t) \), defining the specific set of four local consecutive nodes involved in the weighting algorithm, therefore \( i \in \mathbb{Z}[0, n] \). The parameter \( h \) is the parametric variable defining distance along this local spline segment, and hence \( h \in \mathbb{R}[0, 1] \). The spline value \( s \) at some position \( h \) along a 4-node spline section can therefore be calculated by multiplying a parametric vector with the spline weighting matrix \((W)\), and the specific spline node values \((X_F)\) determined by the value of \( i \).

\[
W = \frac{1}{6} \begin{bmatrix}
-1 & 3 & -3 & 1 \\
3 & -6 & 3 & 0 \\
-3 & 0 & 3 & 0 \\
1 & 4 & 1 & 0
\end{bmatrix}
\]  

(3.9)

\[
s = S(X_F, i + h) = \begin{bmatrix}
h^3 & h^2 & h & 1
\end{bmatrix} WX_F \begin{pmatrix}
i \\
\vdots \\
i + 3
\end{pmatrix}
\]  

(3.10)

The gradient of the spline at this point \((\partial s/\partial t)\) can similarly be found by including a simple differentiation matrix. Curvature \((\partial^2 s/\partial t^2)\) can be determined in a similar way.

\[
\frac{\partial s}{\partial t} = S'(X_F, i + h) = \begin{bmatrix}
3h^2 & 2h & 1 & 0
\end{bmatrix} WX_F \begin{pmatrix}
i \\
\vdots \\
i + 3
\end{pmatrix}
\]  

(3.11)

### 3.3.3 Spline Fitting

As previously mentioned, direct edge extraction from imagery will produce data represented via a pixel list. This is an undesirable representation scheme, and as such it is beneficial to re-classify these curves into a more useful spline based formulation. This is performed through data-fitting of a spline to the raw
The general form of the algorithm to fit a number of points to a spline, involves starting with an estimate of the feature spline node sequence ($S_t$) of length $n + 1$. This estimate is then refined through a least-squares regression, in order to generate the fitted spline nodes $S_f$. This can be performed by identifying the closest points on this spline to each of the points to be fitted, and the associated jacobians relating them to the test spline nodes:

$$\Delta Y = Y_s - Y_p$$  \hspace{1cm} (3.12)
$$\Delta S = \left( \frac{\partial Y_p}{\partial S_t} \frac{\partial Y_p}{\partial S_f} \right)^{-1} \frac{\partial Y_p}{\partial S_t} \Delta Y$$  \hspace{1cm} (3.13)
$$S_f = S_t + \Delta S$$  \hspace{1cm} (3.14)

where $Y_s$ and $Y_p$ are the points to be fitted, and the closest on-spline association respectively. The test spline $S_t$, can be estimated by interpolating node positions along the detected edge pixel list at a frequency lower than that of the sensor measurements. The resulting spline node distribution from this fitting algorithm may then be used as a spline estimate for an iterated fitting process.

The spline fitting process requires knowledge of the closest point on a spline to each data point. For splines of greater than one dimension, this must be performed numerically.

### 3.3.4 Closest Point On Spline

Determining the closest point on a spline to any arbitrary point is a complex process that cannot be performed algebraically. However using a three-step numerical process, a good approximation can be determined:

**Distance to regular points along spline:**

The closest spline point can be approximated by calculating the distance to a number of points along the spline, spaced at regular intervals of length $\epsilon$. This results in a coarse estimate of the closest point, $t_r$. 

59
Quadratic–Fit Approximation:

A more accurate, algebraic approximation can now be performed by fitting a parabolic function to three points on the spline around the coarse estimate \( t_r \) determined in the previous section. The parabolic minimum can then be easily determined, resulting in a more accurate approximation \( t_{quad} \), of the closest parametric point on the spline.

\[
d_0 = |S(X_F, t_r) - H_s| \quad (3.15)
\]
\[
d_+ = |S(X_F, t_r + \epsilon) - H_s| \quad (3.16)
\]
\[
d_- = |S(X_F, t_r - \epsilon) - H_s| \quad (3.17)
\]
\[
A = \frac{2\epsilon(d_- - d_0) + \epsilon(d_+ - d_-)}{2\epsilon(t_r^2 - (t_r - \epsilon)^2) + \epsilon((t_r + \epsilon)^2 - (t_r - \epsilon)^2)} \quad (3.18)
\]
\[
B = d_0 - d_- - \frac{A}{\epsilon}(t_r^2 - (t_r - \epsilon)^2) \quad (3.19)
\]
\[
t_{quad} = -\frac{B}{2A} \quad (3.20)
\]

where \([d_0, d_+, d_-]\) are the distances between the points on the spline at parametric positions \([t_r, t_r + \epsilon, t_r - \epsilon]\) respectively, and the new sensor measurement \( H_s \). \( A \) and \( B \) are the quadratic fit coefficients, following the well-known parabolic model, \( y = Ax^2 + Bx + C \).

Newton’s Method Refinement:

Once this reasonable approximation has been determined, accuracy can be increased further by the iterative calculation of the spline slope and curvature around the best point, coupled with using Newton’s method to further minimise the distance. This process can be repeated until convergence.
\[ \Delta s = S(X_F, t_i) - H_s \]  

\[ \partial s = S'(X_F, t_i) \]  

\[ \partial^2 s = S''(X_F, t_i) \]  

\[ t_{[i+1]} = t_{[i]} - \frac{\Delta s \cdot \partial s}{\partial s \cdot \partial s + \Delta s \cdot \partial^2 s} \]  

where the process is initialised at \( t_{[1]} = t_{quad} \).

### 3.3.5 The Non-Uniform Rational B-Spline (NURBS)

The non-uniform rational basic spline, or NURBS is a generalisation to the b-spline allowing surfaces or volumes to be represented. This requires multiple parametric variables to define a particular point. Effectively, a cubic NURBS defines new temporary nodes successively in different dimensions, which in turn define new points using the next parametric variable. A 2D surface NURBS would therefore require 16 coefficients, which will generally form a 4 by 4 grid, i.e. cubic splines in each direction.

\[ n = N(X_{2F}, i + h, j + g) \]  

\[ = S \begin{bmatrix} S(X_{2F}(j), i + h) \\ S(X_{2F}(j + 1), i + h) \\ S(X_{2F}(j + 2), i + h) \\ S(X_{2F}(j + 3), i + h) \end{bmatrix}, g \]  

where \( X_{2F}(j) \) is the \( j \)-th column of the 2D coefficient matrix \( X_{2F} \). Parametric variables \( j \) and \( g \) are the equivalents of \( i \) and \( h \) respectively, in the second dimension of the NURBS surface.

### 3.3.6 Spline Curve Association

Robust spline based association techniques predominantly require the evaluation of gradient, curvature and length information of these spline curves. In the case of one-dimensional splines, many of these parameters are trivial. However the addition of extra dimensions to these curves increases complexity.
significantly. For this reason, the dimensionality involved in the curve association will be limited to two.

For TAN, this results in consideration of curve movement in north and east, however vertical displacement is ignored.

The gradient $\nabla s$, or curve orientation in the ground plane at a specific point on the curve can be determined algebraically through the use of the spline point derivative equation outlined in section 3.3.2:

\[
\begin{align*}
\partial N &= s'(X_N, t) \quad (3.27) \\
\partial E &= s'(X_E, t) \quad (3.28) \\
\nabla s &= \tan^{-1}(\partial E/\partial N) \quad (3.29)
\end{align*}
\]

where $X_N$ and $X_E$ are the spline node coefficients for position north and east respectively, and $t$ is the parametric variable defining a point on the spline. $\partial N$ and $\partial E$ are the first-order spline derivatives in the north and east dimensions.

The local curvature $\nabla^2 s$ of a point on the spline can be calculated using the second order spline derivatives (see appendix 10.1):

\[
\begin{align*}
\partial^2 N &= s''(X_N, t) \quad (3.30) \\
\partial^2 E &= s''(X_E, t) \quad (3.31) \\
\nabla^2 s &= \frac{\partial N \cdot \partial^2 E - \partial E \cdot \partial^2 N}{(\partial N^2 + \partial E^2)^{3/2}} \quad (3.32)
\end{align*}
\]

where $\partial^2 N$ and $\partial^2 E$ are the second-order derivatives of the spline in the north and east dimension respectively.

Information related to the nature of a spline curve between two points can also be evaluated algebraically, although this is complicated by the piecewise nature of the spline. The cumulative curvature between two points can be determined in a naïve fashion by calculating the gradient of the two points separately, and taking the difference. This will result in errors should any loops in the curve be present between these two points. Tolerance to loops in the spline can be provided through determining the angular difference between the $h = 0$ and $h = 1$ positions of each spline segment between the two target points. These differences may then be safely accumulated, as it is not possible for an individual spline
section to exhibit an angular separation of greater than $180^\circ$.

The curved perimeter distance $d$ between two points on a spline can also be evaluated algebraically. This also requires intermediate spline segments to be analysed separately, with the total length resulting from the accumulation of each piece. The algebraic expression for the 2-dimensional perimeter distance segment $d_n$ is highly complex, and as such is presented in integral form:

$$d_n = \int_0^h s'(X_N, n + h) \cdot s'(X_E, n + h) \cdot dh$$

(3.33)

$$d = \sum_{n=0}^{i} d_n$$

(3.34)

where parametric variable $h$ defines the distance along the local target spline segment, and $n$ outlines which segment is presently being considered. The combination of spline perimeter distance, gradient change and local curvature can be used to assist in unique, identifiable classification of spline sections.

One further algebraic method of analysing splines is the identification of inflection points. Each spline section must be analysed separately due to the piecewise nature of the spline. These inflections can be found through the algebraic determination of a value of $h$ which is the solution of $\nabla^2 s = 0$. Should this resulting value of $h \in \mathbb{R}[0, 1]$, this spline section contains an inflection point located at $h$. Otherwise, should $h \notin \mathbb{R}[0, 1]$, the section contains no inflection.

### 3.4 Parameter Estimation

Mathematical models can usually be used to describe the behaviour of physical systems. In the case that the specifics of these models are not known, often they can be inferred from observed behaviour. The use of observations to reconstruct unknown terms involved in a mathematical model is known as parameter estimation. A number of different algorithms exist to perform parameter estimation, based on the complexity of the system model, which is to be regressed. These can range from measurements of a single scalar parameter where an average or weighted mean will suffice, to nonlinear multi-parameter systems, which require more involved methods to solve.
3.4.1 Solution Covariance

Often when a number of measurements are taken of a particular parameter, any noise or other uncertainties in the system, will cause these measurements to be randomly distributed about the true value. It is therefore beneficial (and trivial) to employ all these individual measurements to estimate this true value by taking the average, or mean. Once this is calculated it is also beneficial to estimate the level of certainty in this solution, which can be performed by deriving the standard deviation $\sigma$, or the variance $\sigma^2$. This method can also be generalised to hold true for multi-dimensional measurements. Uncertainty can be quantified by generating a covariance matrix, which combines the individual variances of each sensor measurement dimension, with the cross-coupling dependencies between them. The terms in the covariance matrix can be calculated through the sum of the mean residuals:

$$P_{i,j} = \frac{1}{N - 1} \sum_{n=1}^{N} (x_{i,n} - \bar{x}_i)(x_{j,n} - \bar{x}_j)$$ (3.35)

where $N$ is the number of measurements, $x_{i,n}$ and $x_{j,n}$ are the $i$ and $j$ dimensions of the $n$-th measurement, $\bar{x}_i$ and $\bar{x}_j$ are the $i$ and $j$ dimensions of the mean solution $\bar{x}$.

3.4.2 Ordinary Least Squares

Ordinary least squares regression is a method of solving a set of equations describing an overdetermined system. This method finds a solution which minimises the squares of the residuals, between the determined solution and the individual system equations. This is performed by considering the linear system $\vec{b} = A\vec{x}$, where the length of known parameters $\vec{b}$ is greater than the length of unknown parameters $\vec{x}$. The vector $\vec{x}$ can therefore be calculated through matrix division:

$$\vec{x} = (A^T A)^{-1} A^T \vec{b}$$ (3.36)

Should the equations involved be nonlinear such that $\vec{b} = f(\vec{x})$, the matrix $A$ may still be found through algebraic differentiation, or numerical perturbation. A nonlinear least squares solution will also generally require an iterative process to solve. This involves determining an initial guess of $\vec{x}$, and using linearised least squares to refine this guess until convergence. This method however, may result in a
divergent process in some cases based on the quality of the initial guess, as the iterative method may only converge to a local minimum. Care must be therefore be taken to identify divergent solutions, and re-evaluate initial conditions should this occur.

3.4.3 Weighted Least Squares

Weighted least squares is an extension of the ordinary least squares algorithm, which allows each of the known parameters $\vec{b}$ to be given more or less importance in the regression. A weighting matrix $W$ is defined as an exclusively diagonal, square matrix of terms outlining the desired weighting of each term in $\vec{b}$. This method can be used to determine an optimal solution to known parameters which exhibit varying levels of certainty ($\sigma$), by setting the weighting terms to $w_{i,i} = 1/\sigma_i^2$. The least squares regression equation therefore becomes:

$$\bar{x} = (A^TWA)^{-1}A^TW\vec{b} \quad (3.37)$$

3.4.4 Generalised Least Squares

Generalised least squares is a further generalisation of the least squares algorithm, where the weighting matrix $W$ contains correlation information between terms. If it is known that the parameters in $\vec{b}$ not only have different certainties, but that those certainties are cross-coupled, a covariance matrix $P$ can be derived. This allows the least squares solution to reflect this coupling, by defining a new, non-diagonal weighting matrix, $W = P^{-1}$.

3.4.5 Chi-Square Test

In this section, various methods have been shown which allow a number of measurements to be regressed into a representative solution. This solution also exhibits an uncertainty bound, which encompasses the original measurements. Once these values have been determined, it is possible to ascertain if any new measurement belongs to this set. This can be performed using a chi square, or $\chi^2$ - test. The difference (innovation) between the set mean $\bar{x}$, and the new measurement $x$, is used to evaluate the similarity
with the original data. This similarity also depends on the solution covariance $P$, and the measurement variance $P_s$:

$$I = x - \bar{x}$$  \hspace{1cm} (3.38)

$$\chi^2 = I^T(P + P_s)^{-1}I$$  \hspace{1cm} (3.39)

where $I$ is the innovation. The larger $\chi^2$ is, the less likely the new point belongs to the original data. It is therefore prudent to choose a threshold for this value.

The chi-square test can be used as a crude data association technique, or may also be used to detect errors in other data association methods, or for other aspects of fault detection.

### 3.5 Extended Kalman Filter

Least squares methods of parameter estimation require that all information is available simultaneously, and therefore all this information can be used in the estimation process all at once. For time varying systems, this is less useful, as measurements at any particular moment, may be insufficient to provide a complete solution estimate. The extended Kalman filter is a time varying extension of the generalised least squares estimator where measurements taken at different intervals can be combined to produce a state estimate. This is performed via a two step process, repeating as system time advances. The first step is a prediction, which produces the relationship between the measurements and states at different points in time. The second step is the Kalman update, which links measurements at a specific time with the present state estimate at that same moment. This process is therefore to make an initial guess, and then refine that guess with new information. To that end, state and uncertainty terms $X$ and $P$ are defined as $X^-$ and $P^-$ during the post-prediction initial guess state. Once updated with new information, $X^+$ and $P^+$ terminology is used, denoting the increase in accuracy.
3.5.1 Prediction Step

The prediction step of the EKF is derived from a system model that describes the evolution of the system states, with respect to time. This system model $f(X, U)$ will generally be a function of the present state $X$, as well as any extra input terms $U$. In a typical aerial navigation scenario, the state $X$ will represent the complete vehicle pose, whereas $U$ may denote inertial measurements or control surface positions. This model may therefore be used to propagate the current vehicle state $X_t^+$ to a future state $X_{t+1}^-$ over a time separation period of $\Delta t$. Any inertial measurements $U_t$ made during this time are used to assist this propagation, typically via an integration process.

$$\dot{X}_t = f(X_t^+, U_t) \quad (3.40)$$
$$X_{t+1}^- = \Delta t \dot{X}_t + X_t^+ \quad (3.41)$$

The Kalman prediction step also considers the current uncertainty of each state parameter, as well as any cross-coupling between terms. This is performed through tracking of a covariance matrix $P$. The evolution of this matrix is dependent on the sensitivity of each state parameter to each other parameter ($F$), as well as sensitivities to any of the extra input terms ($G$). These sensitivities are calculated through the linearised derivatives about the present state either through numerical perturbation, or preferably algebraic means.

$$F = \frac{\partial X_{t+1}}{\partial X_t} \quad (3.42)$$
$$G = \frac{\partial X_{t+1}}{\partial U} \quad (3.43)$$
$$P_{t+1}^- = FP_t^+ F^T + GQG^T \quad (3.44)$$

where $Q$ is the uncertainty covariance of any extra input terms, such as inertial measurements.

The result of this propagation of covariances is that state uncertainties will always increase during the Kalman prediction step. This is due to modelling errors, current uncertainties in state parameters compounding, as well as uncertainties in any prediction-related measurements, such as from an IMU or control surface transducers. In order to limit the increase in uncertainty, the Kalman update step must be used.
3.5.2 Update Step

The update step of the Kalman filter is the mechanism where new information pertaining to the state parameters is used to improve the estimates. For this to be performed, mathematical models are required which describe the expected values of sensor measurements based on the current state parameters. This could consist of dynamic pressure estimates derived from the velocity of the vehicle. Other examples could be predictions of VOR/DME estimates considering known positions of ground installations and the position estimate of the vehicle. These sensor predictions $Y_p$ are compared to the true sensor readings $Y_s$ to result in an innovation $\Delta Y$, or difference. Sensitivity jacobians can be derived from these sensor models, describing the sensitivity of the measurements to the state parameters ($C$). This information, along with the uncertainties of the measurements themselves $R$, may be used to both refine the state parameters and reduce their uncertainties. The Kalman update step is outlined below:

\[
\Delta Y = Y_s - Y_p \tag{3.45}
\]
\[
C = \frac{\partial Y_p}{\partial X} \tag{3.46}
\]
\[
S = CP_{t+1}C^T + R \tag{3.47}
\]
\[
K = P_{t+1}C^TS^{-1} \tag{3.48}
\]
\[
X_{t+1}^+ = X_{t+1}^- + K\Delta Y \tag{3.49}
\]
\[
P_{t+1}^+ = (I - KC)P_{t+1}^- \tag{3.50}
\]

where $\Delta Y$ is the Kalman innovation vector, $Y_s$ are sensor measurements, and $Y_p$ are the predicted measurements. $S$ denotes a sum of the uncertainties, which is used to determine $K$, the Kalman weighting matrix, usually referred to as the Kalman Gain.

3.6 Simultaneous Localisation and Mapping

As outlined above in the previous section, the Kalman filter involves the estimation of a static number of state parameters, which may evolve over time. More recently, this idea has been extended to allow new state parameters to be initialised during operation of the Kalman filter, which are then estimated under the same process as any original states. This method is known as simultaneous localisation and mapping.
The traditional EKF parameter estimator operates to localise the vehicle, while the initialisation of new features will result in a mapping process. The major change to this system is the inclusion of initialisation algorithms, however some minor modifications of the Kalman prediction and update steps must also be considered.

3.6.1 Feature State Initialisation

The major advantage of the SLAM process over the standard EKF is the addition of on-the-fly state initialisation. Generally, these states will describe newly identified features. In order to add these new features to the Kalman state vector, they must be characterised, and their uncertainties evaluated. Characterisation of these features involves the initial evaluation of any feature states $X_f$ which are to be estimated, such as position. The uncertainty of these feature states $P_f$ must also be derived. In order to perform these initial estimates of the feature state and uncertainty, mathematical models of the involved sensors are required.

Consider the model $Y_s = g(X, H_f)$, which describes the expected sensor measurement $Y_s$ of a feature based on the present vehicle state $X$, as well as the characteristics of the feature $H_f$ (such as its position). This is the same model as used for the standard EKF update step. From this, an inverse sensor model can be derived $H_f = h(X, Y_s)$ which describes the expected feature state values, which would result in a particular sensor measurement. This inverse model can also be used to generate jacobians describing feature state sensitivity to the vehicle state ($\partial H/\partial X$), as well as to the sensor measurement ($\partial H/\partial Y$). This information can therefore be used to resolve the expected uncertainty of the feature state based on a preliminary measurement:

$$P_f = \frac{\partial X_{fn}}{\partial X_v} P_v \frac{\partial X_{fn}}{\partial X_v}^T + \frac{\partial X_{fn}}{\partial Y_p} R \frac{\partial X_{fn}}{\partial Y_p}^T \quad (3.51)$$

where $P_v$ is the predicted uncertainty of the vehicle, and $R$ is the uncertainty of the sensor measurement. This uncertainty, combined with the estimated values of the feature state itself, can then be appended to...
the Kalman state vector and covariance matrix:

\[
X = \begin{bmatrix}
X_v & X_{f_1} & \cdots & X_{f_i} & \cdots & X_{f_j} & \cdots & X_{f_n}
\end{bmatrix}^T
\]

\[
P = \begin{bmatrix}
P_{v^2} & P_{vf_1} & \cdots & P_{vf_i} & \cdots & P_{vf_j} & \cdots & P_{vf_{n-1}} & \frac{\partial X_{f_n}}{\partial X_v}^T
P_{f_1v} & P_{f_1^2} & \cdots & P_{f_1f_i} & \cdots & P_{f_1f_j} & \cdots & P_{f_1f_{n-1}} & \frac{\partial X_{f_n}}{\partial X_v}^T
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots
P_{f_jv} & P_{f_jf_1} & \cdots & P_{f_jf_i} & \cdots & P_{f_jf_j} & \cdots & P_{f_jf_{n-1}} & \frac{\partial X_{f_n}}{\partial X_v}^T
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots
\frac{\partial X_{f_n}}{\partial X_v} P_{v^2} & \frac{\partial X_{f_n}}{\partial X_v} P_{vf_1} & \cdots & \frac{\partial X_{f_n}}{\partial X_v} P_{vf_i} & \cdots & \frac{\partial X_{f_n}}{\partial X_v} P_{vf_j} & \cdots & \frac{\partial X_{f_n}}{\partial X_v} P_{vf_{n-1}} & \frac{\partial X_{f_n}}{\partial X_v} P_{f_n^2}
\end{bmatrix}
\]

where \(X_v\) is the vehicle state estimate, and \(X_{f_1} \cdots X_{f_{n-1}}\) are the state estimates of any previously initialised features. Similarly \(P_{v^2}\) and \(P_{f_1^2} \cdots P_{f_{n-1}^2}\) are the variances of the vehicle and any previously initialised features. \(P_{vf_i} \cdots P_{vf_{n-1}}\) are the cross-coupling terms between the vehicle and feature states.

Note that the covariances between different feature states are presented in the form \(P_{f_i f_j}\) (i.e. the variance sensitivity of \(f_i\) to perturbations in \(f_j\)).

### 3.6.2 Kalman Prediction Modification

In the case of SLAM, the Kalman prediction step remains identical to that of the EKF, however it is important to make a few observations. The EKF prediction step propagates the current states to a future point in time, using a pre-determined mathematical model of how these states evolve. In the case of SLAM, this propagation is also necessary for feature states, therefore a model of how these feature states evolve with time is required. Fortunately, for most instances of SLAM viewed features can be assumed to be stationary, and as such the resulting process model is trivial (i.e. \(\dot{x}_{f_i} = 0\)). In the case of moving features, motion models must be used. This is known as moving object tracking SLAM, or SLAM-MOT.

Upon reviewing the state vector and covariance matrix for SLAM (Eqn. 3.52) it becomes clear that when many features have been initialised, these elements can become very large. From the covariance propagation equation (Eqn. 3.44), it can therefore be seen that the computational complexity scales
poorly with number of state parameters. That is to say, computation time increases geometrically \((O(n^3))\) with the number of states. Fortunately, in the case where features are assumed to be stationary, this can be simplified. This is because only the vehicle state parameters evolve over time, whereas features are stationary. As such, only the vehicle state uncertainties and the cross-correlations specifically related to the vehicle are affected. The Kalman prediction update step can therefore be rewritten:

\[
P_{t+1}^- = \begin{bmatrix}
FP_{v^2} F^T & (FP_{f_1...n^v} v^T) \\
FP_{f_1...n^v} & P_{f_1...n^v}
\end{bmatrix} + \begin{bmatrix}
GQG^T & 0 \\
0 & 0
\end{bmatrix}
\]

(3.54)

where \(P_{v^2}\) is the vehicle state quadrant of \(P_{t}^+\), \(P_{f_1...n^v}\) the combined feature state quadrant, and \(P_{f_1...n^v}\) the cross-correlation of all feature states to the vehicle (i.e. the lower left quadrant of \(P_{t}^+\)). Note that \(f_1...n\) represents all of the feature states. It can therefore be seen that this method reduces the computational complexity to linear scaling \((O(n))\), with the number of feature states.

### 3.6.3 Kalman Update Modification

Similar to the Kalman prediction, the update step in SLAM remains fundamentally unchanged to the standard EKF formulation. It is, however, important to note that the addition of feature states results in some practical differences, despite being mathematically identical. This is because the measurement model in the standard EKF is only dependent on the vehicle state parameters, whereas with SLAM, dependence on feature state parameters is also obtained. Therefore, it is convenient to rewrite the sensor model \(Y = g(X, H_f)\) for the EKF to take the form \(Y = g(X_v, X_f)\). In the standard EKF formulation, \(H_f\) is assumed to be known perfectly, whereas for SLAM the feature state \(X_f\) is estimated, with an associated variance.

At any point that an initialised feature is re-observed, the Kalman update step is evaluated through the calculation of a predicted measurement, and the associated state-to-measurement Jacobian. In the case of SLAM, this Jacobian will contain sensitivities to both the vehicle states, as well as the viewed feature states. Sensitivities to other features will be zero. This is shown below:

\[
\frac{\partial Y_p}{\partial X} = \begin{bmatrix}
\frac{\partial Y_p}{\partial X_v} & 0 & \cdots & \frac{\partial Y_p}{\partial X_f} & \cdots & 0
\end{bmatrix}
\]

(3.55)
where \( X_{f_n} \) are the states of the viewed feature. Once this Jacobian matrix has been evaluated, the EKF update step may be performed normally, as outlined in section 3.5.2.

### 3.7 State and Sensor Models

As outlined above in the previous two sections, the EKF and SLAM require prior knowledge of a number of mathematical models. These include models of the state evolutions with time, as well as models for all sensors which provide information for fusion. For the purposes of the work proposed in this thesis, these models can be separated into a prediction model, visual update models and non-visual update models.

#### 3.7.1 Vehicle Process Model

Part of the EKF–SLAM algorithm is the prediction step, which involves using a process model to estimate the future position of the vehicle. This can be done in a general 6–DOF case by using a rigid-body assumption. Linear accelerations and rotation rate measurements obtained from an IMU can be employed to govern the evolution of the state, in the vehicle body frame. The rate of change of the vehicle states can be determined using equations 3.56 - 3.58:

\[
\dot{X}_{vel} = U_{acc} + C_{bn} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \begin{bmatrix} 0 & -w & v \\ w & 0 & -u \\ -v & u & 0 \end{bmatrix} U_{rot} \tag{3.56}
\]

\[
\dot{X}_{Att} = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix} \frac{U_{rot}}{2} \tag{3.57}
\]

\[
\dot{X}_{Pos} = C_{nb} X_{vel} \tag{3.58}
\]

\[
X_{t+1} = X_t + \Delta t \dot{X} \tag{3.59}
\]

where \( U_{acc} \) and \( U_{rot} \) are the accelerations and rotation rates in body frame returned from the 6-axis IMU. \( X_{vel} = [u, v, w] \) are the vehicle velocity estimate components in body frame. The direction cosine matrix
(DCM) \( C_{bn} \) and quaternion attitude components \([q_0, q_1, q_2, q_3]\) describe the local vertical, local horizontal (LVLH) earth navigation frame to body frame rotational transform. The quaternion components \( q_1, q_2 \) and \( q_3 \) define the cartesian rotation vector, and \( q_0 \) the magnitude and direction of the rotation. \( \Delta t \) is the IMU update period.

These equations can then be used to determine the process model Jacobians required to predict the state uncertainties represented in the covariance matrix \((P)\). This involves partial differentiation with respect to each state, as well as the IMU measurements.

### 3.7.2 Observation Models

The update step of the Kalman filter is based on the comparison between sensor measurements and predictions, which result in sensor innovations. These predictions are based on the current estimated state values, derived from a mathematical model of the sensor. This model is also used to generate the partial derivative matrices, known as Jacobians, which determine how state estimates evolve to minimise new innovations. These models may also be used for the mapping section of the SLAM algorithm, allowing the direct estimation of state parameters from sensor measurements.

#### Visual Updates

An observation model which replicates the optics of a digital camera is required to estimate visual feature updates. The information available to the model is: the present pose of the vehicle \((X)\), the angular offset between the vehicle body and the camera bore-sight \((C_{cb})\), and also the position of the viewed feature \((H_f)\). The model must predict both the horizontal \((\chi)\) and vertical \((\lambda)\) position of a feature in the camera frame. The range \((R)\), i.e. the interval distance between feature and vehicle can also be determined from the available information. These resulting parameters allow the sensor estimation of point features.

This work also deals with curve features, and is not limited to the estimation of points. It is therefore beneficial to extend this model to the prediction of the apparent orientation of curve points in the camera frame. This can be performed by also considering the partial derivatives of a point on the curve, relative to distance along the curve \((\partial H / \partial t)\). Apparent orientation information may then be determined both in
the camera image plane (\(\gamma\), curve gradient in image), as well as out of the image plane (\(\epsilon\), the rate of change of range with distance along the curve). Figure 3.2 graphically outlines the definitions of these parameters.

This visual sensor prediction can be performed using a simple pinhole camera model. This model is extended to include gradient terms, allowing the estimation of curve orientation.

![Figure 3.2: Definition of angular frame positions \([\chi, \lambda]\) (Eqn. 3.62) and curve slope angles \([\gamma, \epsilon]\) (Eqn. 3.64) in camera frame (a) as well as with a 3D representation (b) also showing range \(R\) and the temporary orthogonal camera axis components \([C_x, C_y, C_z]\).](image)

**Camera Model** The pinhole camera model involves a linear approximation of the optics of a camera. Fortunately, as the optical lenses in cameras are generally designed to produce a linear distribution, this approximation is sufficiently accurate. This linearisation is performed in an orthogonal axis (\(C\)) aligned to the camera, and therefore requires coordinate rotations between Earth frame and this camera frame:

\[
\Delta H = H_f - X_{Pos} \quad (3.60)
\]

\[
\Delta C = C_{cb}C_{bn}\Delta H \quad (3.61)
\]

where \(\Delta H\) is the Earth frame position difference between known feature (\(H_f\)) and vehicle (\(X_{Pos}\)) positions. Rotation matrices \(C_{bn}\) and \(C_{cb}\) describe the Earth navigation to body, and body to camera DCMs respectively.

Once the relative position of the feature in the camera axis is known, a linear estimate of the angular position of the feature in the camera image can be made. This relative position also allows the range to
the feature to be estimated:

\[ \hat{Y} = \begin{bmatrix} R \\ \chi \\ \lambda \end{bmatrix} = \begin{bmatrix} \sqrt{\Delta C_x^2 + \Delta C_y^2 + \Delta C_z^2} \\ \Delta C_y/\Delta C_x \\ \Delta C_z/\Delta C_x \end{bmatrix} \]  \hspace{1cm} (3.62)

where \( \Delta C_x, \Delta C_y \) and \( \Delta C_z \) are the orthogonal components of \( \Delta C \), i.e. distance forward, left and down from the camera centre.

The apparent orientation of a curve at a point, may similarly be estimated from a curve orientation unit vector \( \partial H \) in Earth frame. This can be rotated into the camera space frame, to give a new unit vector \( \partial C \). Gradient angles may then be determined:

\[
\begin{bmatrix} \partial C \\ \gamma \\ \epsilon \end{bmatrix} = \begin{bmatrix} C_{cb}C_{bn}\partial H \\ -\arctan\left(\frac{\partial C_z}{\partial C_y}\right) \\ \arctan\left(\frac{\partial C_x}{\sqrt{\partial C_y^2 + \partial C_z^2}}\right) \end{bmatrix} \]  \hspace{1cm} (3.63)

Jacobian sensitivity matrices may then be developed using these transform equations. To avoid over-complication of the resulting derivatives, this is performed in parts. Firstly, the derivatives involved in the determination of the camera space frame from LVLH frame are ascertained, giving \( \partial H_C/\partial H_E \). Then, the derivatives involved in calculating angular position and range may be found, giving \( \partial Y/\partial H_C \). Using this information, the chain rule may be employed to determine the required Jacobian, \( \partial Y/\partial H_E \).

\[
\frac{\partial H_C}{\partial H_E} = C_{cb} \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_0q_3 + q_1q_2) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_0q_1 + q_2q_3) \\ 2(q_0q_2 + q_1q_3) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \]  \hspace{1cm} (3.65)

\[
\frac{\partial Y}{\partial H_C} = \begin{bmatrix} H_{cz}/R & H_{cy}/R & H_{cx}/R \\ -H_{cy}/H_{cz}^2 & 1/H_{cz} & 0 \\ H_{cx}/H_{cz}^2 & 0 & -1/H_{cz} \end{bmatrix} \]  \hspace{1cm} (3.66)

\[
\frac{\partial Y}{\partial H_E} = \frac{\partial Y}{\partial H_C} \frac{\partial H_C}{\partial H_E} \]  \hspace{1cm} (3.67)
The sensitivity of the visual measurements to the vehicle state may be determined in a similar way:

\[
\frac{\partial H_C}{\partial X_{Att}} = C_{cb} \left[ \frac{\partial (C_{be} H_E)}{\partial q_0} \frac{\partial (C_{be} H_E)}{\partial q_1} \frac{\partial (C_{be} H_E)}{\partial q_2} \frac{\partial (C_{be} H_E)}{\partial q_3} \right] \quad (3.68)
\]

\[
\frac{\partial H_C}{\partial X_{Pos}} = - \frac{\partial H_C}{\partial H_E} \quad (3.69)
\]

\[
\frac{\partial Y}{\partial X} = \begin{bmatrix}
0 & \frac{\partial Y}{\partial C_{be}} \frac{\partial H_C}{\partial X_{Att}} & \frac{\partial Y}{\partial H_C} \frac{\partial H_C}{\partial X_{Pos}}
\end{bmatrix} \quad (3.70)
\]

where the sensitivities of \( H_C \) to the quaternion attitude is quite complex, and therefore has been presented in partial differential form. Also note that the sensitivity of the visual measurement to the vehicle velocity is zero.

The sensitivity of the local curve gradients may also be determined in a similar manner:

\[
\frac{\partial G_C}{\partial X_{Att}} = C_{cb} \left[ \frac{\partial (C_{be} G_E)}{\partial q_0} \frac{\partial (C_{be} G_E)}{\partial q_1} \frac{\partial (C_{be} G_E)}{\partial q_2} \frac{\partial (C_{be} G_E)}{\partial q_3} \right] \quad (3.71)
\]

\[
A = \partial C_y^2 + \partial C_z^2 \quad (3.72)
\]

\[
B = \partial C_x^2 + \partial C_y^2 + \partial C_z^2 \quad (3.73)
\]

\[
\frac{\partial G_Y}{\partial G_C} = \begin{bmatrix}
0 & \frac{\partial C_z}{A} & -\frac{\partial C_y}{A} \\
\sqrt{A}/B & -\frac{\partial C_y \partial C_z}{B \sqrt{A}} & -\frac{\partial C_x \partial C_z}{B \sqrt{A}}
\end{bmatrix} \quad (3.74)
\]

where \( G_Y = [\gamma, \epsilon]^T \) represents the angular curve orientation, and \( G_C = [\partial C_x, \partial C_y, \partial C_z]^T \) represents the normalised curve direction vector.

**Inverse Camera Model** The pinhole camera model may be algebraically inverted, resulting in a relationship which approximates the location of a feature in world frame, given a position in camera frame and the vehicle pose. The mapping step of the EKF–SLAM algorithm relies on this to initialise new features into the state vector. The resulting relationships are:

\[
\Delta C_x = \frac{R}{1 + \chi^2 + \lambda^2} \quad (3.75)
\]

\[
\Delta C_y = \chi \Delta C_x \quad (3.76)
\]

\[
\Delta C_z = -\lambda \Delta C_x \quad (3.77)
\]
This therefore results in the position interval between vehicle and feature, in the camera space frame. This interval is then rotated into the LVLH world frame:

\[
\Delta H = C_{nb}C_{bc}\Delta C \quad (3.78)
\]

\[
H_f = \Delta H + X_{pos} \quad (3.79)
\]

The orientation of a curve in LVLH frame may be determined in a similar manner:

\[
\partial C_x = -\frac{1}{2} \tan^2 \epsilon + \sqrt{\tan^4 \epsilon + 4 \tan^2 \epsilon} \quad (3.80)
\]

\[
\partial C_y = \frac{\sqrt{(1 - \partial C_z)}}{(1 + \tan^2 \gamma)} \quad (3.81)
\]

\[
\partial C_z = -\partial C_y \tan \gamma \quad (3.82)
\]

\[
\Delta H_f = C_{nb}C_{bc} \begin{bmatrix}
\partial C_x \\
\partial C_y \\
\partial C_z
\end{bmatrix} \quad (3.83)
\]

Jacobian sensitivity matrices of the feature position to changes in vehicle state and sensor measurement can also be determined through partial differentiation, as with the camera transform functions.

**Non-Visual Updates**

Along with the visual updates used in the SLAM process, other standard sensors fitted to airborne platforms may be used to aid the constraint of drift through a more traditional EKF fusion process. A static pressure measurement can be used to improve the vehicle altitude estimate, whereas a magnetometer can be used to constrain attitude. These low cost, robust and passive sensors are already fitted to most aircraft, and already demonstrate widespread use in commercial data fusion systems.

**Magnetometer** A magnetometer measures the magnetic flux vector of the Earth, relative to the vehicle body frame. It can therefore be used to help resolve the vehicle attitude. As this sensor only measures a single vector quantity, magnetometer measurements are insufficient to fully constrain attitude. Instead, measurements from this sensor are invariant to rotations about the direction of the magnetic vector. It is
therefore important to use this sensor as part of a fusion process.

Magnetometer based data fusion requires knowledge of the local magnetic field of the Earth at the
current position of the vehicle. This can be determined using a world magnetic model. The resulting
magnetic flux vector is defined in world LVLH frame, and as such can be converted into a sensor
prediction by co-ordinate rotation into body frame. This can be performed via a DCM rotation matrix
evaluated using quaternion formulations:

\[
C_{be} = \begin{bmatrix}
q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\
2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + q_0q_1) \\
2(q_0q_2 + q_1q_3) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2
\end{bmatrix} \tag{3.84}
\]

\[
M_b = C_{be}M_E \tag{3.85}
\]

where the elements \( q_0 \cdots q_3 \) are the quaternion attitude representations, \( M_E \) is the modelled earth magnetic field in LVLH frame, and \( M_b \) is the magnetic flux estimate in body frame. Jacobian sensitivity
to the vehicle state is determined through finding derivatives of the DCM rotation matrix.

The uncertainty \( R \) of the measurement is a combination of Earth magnetic field modelling errors \( R_E \),
and noise in the sensor itself \( R_Y \). These can be determined through noting the expected accuracy of the
magnetic model used, along with the calibrated or experimentally verified noise characteristics of the
sensor. As magnetic modelling uncertainties will be represented in world frame, it is also necessary to
convert these into body frame using the earth-to-body DCM rotation.

\[
R = R_Y + C_{be}R_EC_{be}^T \tag{3.86}
\]

**Static Pressure** Altitude measurements can be obtained through the use of a static pressure measure-
ment. This is because air pressure varies predictably with altitude, based on the local sea level air pressure.
The current pressure at any altitude can be modelled using standard atmospheric lapse rate equations:

\[
T = T_0 - Lh \tag{3.87}
\]

\[
P = P_0 \left( \frac{T}{T_0} \right)^\eta \tag{3.88}
\]
where $T$ is the current air temperature at vehicle altitude $h$. Also required are the local sea level temperature $T_0$ and pressure $P_0$. The constant $L$ is the atmospheric lapse rate, $R$ is the gas constant and $g$ the acceleration due to gravity.

The sensitivity of this measurement to the vehicle state is for the most part trivial, as altitude is the only parameter that has any effect. The rate of change of pressure with altitude can be determined through differentiation:

$$
\frac{\partial P}{\partial h} = P_0\frac{g}{RT_0} \left( \frac{T_0 - Lh}{T_0} \right) \left[ \frac{2}{\tau R} \right]^{-1}
$$

(3.89)

and used to assemble a jacobian sensitivity matrix for the Kalman update step. This matrix (or vector, as it only consists of a single row) will be predominantly zero, except for the term relating to altitude, i.e. $z$, which evaluates to the aforementioned derivative in Eqn. 3.89.

**Other Common Sensors** Although fusion with dynamic pressure information would also be possible, this was not performed due to concerns about wind. As navigation involves position localisation and velocity estimation in world frame, fusion of velocity measurements relative to the atmosphere can be problematic. This is because this process would require the estimation of the local wind vector, which can change significantly, quite rapidly. The result of this is that any wind estimation will only be valid for short periods of time, and as such the variance of the wind vector estimate will remain large. In fact, absolute wind measurements may only occur when the vehicle velocity is already known, and as such are pointless to a significant degree. Velocity estimates are conceptually better served through the use of visual measurements (which are already in Earth frame) coupled with inertial sensor fusion.

3.8 Optical Flow

Any imagery from a digital video camera will provide information both in a spatial sense, as well as temporally. Treating individual camera frames separately is therefore predominantly ignoring this temporal information. Maximising the use of information from these sensors therefore requires analysis of how objects captured through this imagery change over time, such as their movement. Optical flow is one such method of analysing the temporal information produced by a video stream by detecting the rates of movement of objects through the frame.

3.8.1 Methodology

Optical flow is an umbrella term for a number of different algorithms which are all designed to estimate how parts of an image move between consecutive video frames. These algorithms can generally be separated into two categories:

1. Dense optical flow techniques produce a flow estimate for each pixel in an image. Algorithms in this class may be based on template matching or Fourier methods. These algorithms tend to be computationally expensive, however result in a complete flow field over the entire image.

2. Sparse optical flow techniques produce flow estimates only at certain points in a frame. Algorithms used in this class are more commonly based on point feature classifiers, such as SIFT/SURF, and data association of these features. Sparse methods are generally computationally faster than dense methods, however they do not produce a complete flow field over the entire image.

The performance and applicability of these techniques to any particular system will vary significantly based on the particulars of the target system. For example, collision detection implementations may be better served with sparse methods due to speed, whereas contour mapping may favour dense methods.

3.8.2 Decoupling Relative Motion

In theory, optical flow can be used to track any motion in viewed features or objects. In practice, this process is limited due to coupling of this motion with any motion of the camera. Any motion detected by
optical flow could therefore be the result of any of three kinds of movement:

1. **Motion of the target object** $v_f$ will result in an optical flow reading, depending on the distance between the viewed target object and the camera.

2. **Motion of the camera** $v_c$ will also result in an optical flow reading, also dependent on the distance between the viewed target and the camera.

3. Finally, any rotational motion $r_c$ of the camera will also result in detected optical flow. This flow however is not dependent on any linear distances, only the magnitude of the rotation, relative to the field of view of the camera.

This means the flow measurements are a sum of these three quantities, and as such can be linearly approximated as:

$$\kappa = \frac{A_{fov}}{n_{px}}$$

$$\kappa \vec{F} = \frac{v_f}{R} + \frac{v_c}{R} + r_c$$

where $\vec{F}$ is the flow vector measurement obtained from optical flow and $R$ the separation distance between the target feature object and the camera. $\kappa$ is a scaling factor, relating the horizontal angular field of view $A_{fov}$, to the horizontal number of pixels $n_{px}$. Hence, $\kappa \vec{F}$ has the desired units of radians per second. Note that this is an approximation, as the true motion will be nonlinear in most cases (i.e. should $R$ vary between frames). Furthermore, optical flow will only detect motion in the camera image plane, and as such motion due to changes in $R$ cannot be directly observed.

As any motion detected by an optical flow algorithm may be the result of any one or combination of these three sources, using optical flow as a motion estimator can be problematic. It is therefore incumbent to ensure that any other direct measurements of these parameters are taken (such as inertial rotations) or at the very least estimated to a high accuracy. If this is not possible, different types of motion cannot be separated, and as such the usefulness of the optical flow measurements are severely restricted.
3.9 System Overview and Summary

The background theory outlined in this chapter is used extensively through the development of the work outlined in this thesis. The work itself can be separated into a number of different segments. These areas cover both the construction of a computer vision navigation system, and the methods required to fulfil each part. Most of these segments involve the use of different co-ordinate systems as well as rotations between them, which are outlined in section 3.1. An overview of each segment is presented below:

3.9.1 Inertial Navigation

In the absence of any visually distinct features, a robust navigation system must still be capable of estimating the location of a vehicle. This can be performed through the use of inertial measurements, consisting both of linear accelerations and rotation rates. This estimate can be refined through data fusion with both static pressure and magnetometer measurements. The Kalman prediction and update theory presented in section 3.5 is used in the development of this system, as is the prediction and non-visual sensor models outlined in section 3.7.

3.9.2 Edge Feature Detection

During flight, imagery obtained from a downwards facing video camera shall be processed to determine the location and shape of any visible edges. This involves the use of a watershedding algorithm to segment the image into homogeneous regions. These regions are then examined, with adjacent regions compared, and any regions with high similarity are combined into a single region. A decision is then made based on the colour and texture properties of these regions to determine what ground texture they represent. Boundaries between adjacent regions of differing texture may then be extracted as edge features. The methods required to separate an image into homogeneous sections were outlined in section 3.2.
3.9.3 Visual Spline SLAM

Edge features extracted from the on-board digital video camera can be used to limit inertial drift of the vehicle. This can be performed through the use of the SLAM algorithm, where any new features which have not previously been viewed are initialised as basic spline curves. Through repeated measurements of these edge features, their shape can be refined, and the relative movement of the vehicle estimated. The use of a digital terrain map can be used to infer the range to these features, and therefore, feature initialisation can occur immediately once they are first viewed. Section 3.6 outlines the algorithms necessary to perform SLAM. The manipulation of spline based features requires the definitions reviewed in section 3.3. The construction of a visual SLAM system also relies on the mathematical camera models presented in section 3.7.

3.9.4 Terrain Aided Navigation

Existing satellite and aerial imagery database information can be processed using a similar method to on-board aerial imagery, to extract edge features. These a-priori edge features can be geo-referenced using this database information and stored for use on the visual navigation system platform. During operation of the SLAM algorithm, currently viewed edge features are observed, classified using spline models and their estimates refined. These estimates can be compared to the a-priori spline feature map using gradient and curvature based data association techniques. Positive associations can then be used as absolute position localisation solutions for the vehicle. The use of the EKF-SLAM feature estimates for association is advantageous over the use of raw edges from the image segmentation step for two reasons. Firstly, the estimation process involved in the Kalman filter reduces any noise or the effects of poor texture segmentation solutions. Secondly, depending on the camera field of view and the vehicle altitude, each camera frame may not contain sufficient feature information for a reliable data association. As repeated measurements from a moving vehicle can be used to assemble a much larger feature than any individual frame, the features in the Kalman state estimate are more likely to result in a definitive and unique match. Section 3.3 outlines the background theory involved in performing this update, with the parameter estimation algorithms covered in section 3.4 also used.
3.9.5 Optical Flow Based Odometry

A video camera imagery stream can also be used to constrain integration drift of a vehicle via the use of optical flow. Optical flow is a measurement of the angular velocity of objects viewed by the camera relative to the vehicle. As such, these measurements can be used to estimate the velocity of the camera, and therefore the vehicle, given that a range to the image data can be determined. This can be performed through the assumption that the viewed terrain is stationary. The rotation rates of the vehicle must also be known, which are provided through the use of an IMU. Finally, the range to the terrain is also required, which can be inferred through the use of a DTEM, and the vehicle altitude Kalman estimate. These visual updates require the background theory presented in section 3.8, as well as the camera models outlined in section 3.7.

3.9.6 Visual Terrain Profile Matching

Optical flow measurements may also be used to estimate the distance to, and shape of, the underlying terrain. This estimation is the result of combining the Kalman estimate of velocity with the rotation rate of the vehicle and the optical flow measurements. This terrain profile estimate can be used to determine the absolute position of the vehicle through matching this shape to a known DTEM. Methods can be employed to ensure this matching process is predominantly dependent upon the shape of the terrain, instead of the absolute height of the terrain. This is important as the estimated terrain height accuracy will be dependent on the precision of the vehicle velocity estimate. The inferred variation of the viewed terrain on the other hand will rely much less on this velocity estimate, and will therefore be more robust to velocity errors. As with the visual odometry, this application relies on background theory presented in sections 3.7 and 3.8.
Chapter 4

Image Processing & Feature Detection

The development of an edge-based visual navigation system will rely heavily on the ability of algorithms to extract these edge features from camera imagery. This therefore requires some level of computer vision technology to recognise boundaries between differing ground feature types. This chapter outlines the development of such a computer vision system. As successful visual navigation techniques rely on both update frequency and on accuracy, focus is on developing both computational speed and robustness of the algorithms. The design goals of this section are:

1. Develop a system which identifies human-recognisable edges from aerial imagery, using both colour and texture information.

2. Determine which ground texture types each edge is a boundary between, using a number of human-identifiable ground types.

3. Predict which detected edges are likely to be the result of noise, and remove these features.

4. Consider methods of improving the robustness of the process, and decreasing the effects of changing lighting conditions on the resulting detections.

5. Ensure system is capable of running at, or close to, real time. For the purpose of this work, and considering the typical aircraft dynamic frequency range, real-time is considered to be 5 samples per second.
The work outlined in this chapter aims to achieve these goals, and is presented in the sections outlined below:

**Section 4.1** investigates the typical visual characteristics of ground imagery taken from an aerial platform. These observations are used in the design of the algorithms used to separate and designate ground texture types.

**Section 4.2** outlines the preliminary methods of separating the image into different regions. This is based on the watershedding algorithm, which analyses a gradient magnitude image for concave exclusive regions.

**Section 4.3** describes the use of training data to analyse the texture and colour properties of regions, and designate each a human-identifiable type.

**Section 4.4** presents the methods used to limit the over-segmentation inherent to the watershedding process. This includes filtering steps taken before the watershedding algorithm is performed, as well as recombination processes to eliminate spurious edges.

**Section 4.5** describes the method of extracting pixel list data from the texture segmentation process. This list data is enhanced both through calculation of gradient information of the edges, as well as the boundary textures defining the edge. Filtering and down-sampling techniques are also outlined to minimise data storage requirements, as well as noise.

**Section 4.6** outlines how solar reflections in an image may cause segmentation problems. This section also describes methods of detecting the presence of solar reflections in an image, and of removing any spurious edge features that they may cause.

**Section 4.7** presents samples of the resulting segmented aerial images, outlines an analysis of computation times of the frames, as well as discusses the accuracy of the resulting segmentation.

**Section 4.8** contains a summary of this chapter.
4.1 Ground Appearance

The typical visual characteristics of ground viewed via aerial imagery was investigated. The imagery analysed in this chapter was recorded by the USYD JabLab aerial test platform (section 1.4). The image sequence was obtained during a flight over Lake Burragorang, Warragamba Dam, and surrounding rural areas in NSW, Australia. Different ground texture types were analysed, as were variations within each category. The main candidate types for analysis include water, forest, grass and road.

Various ground texture types can present themselves in a wide range of different colours. An obvious example is water, which can be blue, green or brown, depending on the presence of any sediment or dissolved pigmentation from foliage. Adding to this are the effects of lighting and the atmosphere, which can cause variations in colour content due to direct sunlight or cloud, or wash out colours based on dust, smoke, or any other detriments to visibility. It is therefore clear that colour alone cannot be used to classify the ground texture of viewed terrain. Figure 4.1 demonstrates this, where the colour content of water can be seen to be practically identical to the colour content of a bordering forest. This figure also shows the effects of cloud cover on imagery colour content, which is discussed further in section 4.3.1.

![Figure 4.1: Red, green and blue colour content of trees and water as viewed from the air.](image)

Instead, figure 4.1 shows that the information best suited for distinguishing these two ground types is that of colour variation, or the texture.

Further analysis of aerial imagery clearly demonstrates that even one particular type of ground texture can exhibit greatly varying properties. Figure 4.2 shows a number of different samples of aerial images, containing a number of particular ground types. From this, it is clear that the variation in texture properties
of a single ground type may well be larger than the separations between different types. This is a serious issue for classification algorithms, as mis-classifications become extremely likely.

Figure 4.2: An assortment of aerial imagery obtained from a single flight over Lake Burragorang and Warragamba Dam areas in NSW, Australia.

4.2 Image Segmentation

Segmentation of an image into discrete parts can be performed using a watershed transform. This algorithm is an edge detector based on identifying homogeneous regions in an image. This analysis is performed through the derivation of the gradient magnitude of an image, i.e.

$$M_{i,j} = \sqrt{\left(\frac{\partial I}{\partial u}\right)_{i,j}^2 + \left(\frac{\partial I}{\partial v}\right)_{i,j}^2}$$

(4.1)

where $M_{i,j}$ is the desired gradient magnitude at the $[i, j]$-th pixel. Derivatives $\partial I/\partial u$ and $\partial I/\partial v$ are the horizontal and vertical gradients in the image at these pixel co-ordinates. These derivatives typically can
be determined through discrete methods, i.e.

\[
\left( \frac{\partial I}{\partial u} \right)_{i,j} = \frac{1}{2}(I_{i+1,j} - I_{i-1,j}) \tag{4.2}
\]

\[
\left( \frac{\partial I}{\partial v} \right)_{i,j} = \frac{1}{2}(I_{i,j+1} - I_{i,j-1}) \tag{4.3}
\]

where \( I_{i,j} \) is the value of the pixel at co-ordinates \([i, j]\) in the image. For colour imagery, this process can be performed for each colour channel, with the three resulting gradient magnitudes added together. This summation results in a loss of channel specificity information, however is important for the purposes of computational efficiency. Furthermore, as the goal of this process is to segment the image, the derivation of three distinct segmentation results (one from each colour channel), convolutes the process.

Once the gradient magnitude of the image is determined, the watershed transform algorithm may be applied to this map. A number of different algorithms exist to perform this function, however these processes are based around the same idea. The watershed transform identifies regions where if the gradient magnitude was a relief map, any water droplet falling anywhere within this segment would collect at a single point within this region. The result of this is the separation of the image into discrete regions which each exhibit local concavity, and as such do not contain any detectable edges.

### 4.3 Classifying Textures

In order to efficiently classify regions into different texture types, each region must be reduced to a set of descriptive parameters. These parameters may then be used to compare section properties with those of known ground texture types to achieve a classification match. The choice of parameters which are used as segment descriptors is therefore important, as these descriptors must contain sufficient variation to allow contrast of various ground types.

#### 4.3.1 Texture Parametrisation

One significant hurdle of the visual texture segmentation problem is that of changing lighting conditions. Visual properties of a particular region can change significantly with different viewing times due to the
position of the sun, or the presence of cloud or haze. The predominant cause of this difficulty is changing luminance, as these lighting effects will mostly affect the brightness of colours. They will, however, have less of an effect on the actual colours themselves. This poses a problem for the traditional colour image representation of RGB (red, green, blue), as each channel contains a component of brightness. It is therefore beneficial to use a different colour scheme which separates brightness from the colour content, such as the Lab (lightness, colours a & b) representation. This is preferable to the HSV (hue, saturation, value) colour space as Lab is continuous, whereas the hue channel of HSV is not. Figure 4.3 demonstrates an example of a half-shadowed ground texture, with representations in both RGB and Lab. This figure clearly shows that the Lab colour-space is more tolerant to changing lighting conditions. The colour channels a and b are mostly unaffected by cloud shadows, with the variations occurring almost completely within the lightness (L) parameter.

As demonstrated in section 4.1, colour alone is not sufficient to successfully classify some ground types. For this reason, some indication of texture information is required to characterise how much variation in brightness or colour exists in particular regions. This can be performed in a number of ways, for example using a Gabor frequency filter, or simply determining the standard deviation of a region. These two methods exhibit a number of relative advantages and disadvantages, which will be discussed. Firstly, the Gabor filter provides a method to directly target a particular spatial frequency in an image, for example identifying trees by using a tree-sized kernel. This process however can be rather computationally expensive, especially for large kernel sizes. Also, in order to identify the expected size of a tree, the range to the ground needs to be known. Furthermore, as trees come in a number of different sizes, a number of different Gabor kernel sizes would be required to reliably detect trees. Finally, the
Gabor kernel is based on a low-pass filter, and as such will damage the integrity of edge features and small regions, as demonstrated in figure 4.4. This causes leakage of the properties of one region into any adjacent regions.

![Camera Image, Gabor Response](image)

**Figure 4.4:** Gabor frequency response to an image, and subsequent filter leakage between regions.

The alternative to the Gabor filter is a simple calculation of the standard deviation of brightness in an image. This has the disadvantage of being independent of any frequency information, however this disadvantage also means the algorithm does not need to be re-run multiple times for different size features. Further benefits are that the calculation of standard deviation of a region requires significantly less computation than the Gabor filter response. Finally, as the standard deviation is a property of a discrete region, no inherent filtering is required, and as such this process does not exhibit parameter leakage to adjacent regions.

Analysis of expected ground appearances in section 4.1 also reveal potential issues due to extreme lighting effects. These lighting effects include both over-exposure due to direct sunlight or solar reflections, as well as under-exposure due to shadow. Both of these effects can cause loss of colour information in parts of an image, as well as impacting on the standard deviation of brightness in such areas. This is demonstrated in figure 4.5. For this reason, these areas must be ignored when processing the descriptors of particular regions.

The ramifications of extreme lighting conditions to region characteristics can be reduced by weighting each pixel in a region based on a function of brightness. As both under and over-exposed pixels are undesirable, brightness values approaching both values of zero (black) and one (white) are given a lower weighting than pixels with more reasonable brightness values. Therefore only the pixels with well-conditioned colour information are used in the characterisation of each region. The pixel weighting
can be applied by the use of a weighted extension of the mean and standard deviation expressions:

\[
\bar{p} = \frac{\sum_{i=1}^{n} p_i w_i}{\sum_{i=1}^{n} w_i} \quad (4.4)
\]

\[
\sigma_p = \sqrt{\frac{\sum_{i=1}^{n} w_i (\bar{p} - p_i)^2}{\sum_{i=1}^{n} w_i}} \quad (4.5)
\]

where \( \bar{p} \) is a vector of the mean region properties, and \( \sigma_p \) is the standard deviation. \( p_i \) is a vector of the characteristics of the \( i \)-th pixel in the region, with brightness weighting \( w_i \). \( n \) is the total number of pixels in the region.

![Camera Image](image)

**Figure 4.5:** Demonstration of loss of colour information due to image over-saturation.

### 4.3.2 Parameter Variance

Associations between parameters require an indication of the level of tolerable discrepancy between classifications. In the case of visual colour and texture associations, this can be assumed to be related to the covariance of the region properties. This variance can also be calculated with consideration of the brightness based pixel weightings:

\[
\sigma_{j,k}^2 = \frac{\sum_{i=1}^{n} w_i^2 (\bar{p}_j - p_{j,i})(\bar{p}_k - p_{k,i})}{\sum_{i=1}^{n} w_i} \quad (4.6)
\]

where \( \sigma_{j,k}^2 \) is the \([j, k] \)-th term of the covariance matrix, \( \bar{p}_j \) and \( \bar{p}_k \) are the \( j \) and \( k \) channels of \( \bar{p} \), the weighted property means. \( p_{j,i} \) and \( p_{k,i} \) are the \( j \) and \( k \) channels of the \( i \)-th pixel in the region.

This process will result in the determination of a covariance ellipse which encompasses the expected
pixel values contained in this region. This therefore can be considered as an integral part of the texture association step.

4.3.3 Training Data

In order to determine which parts of an image are different texture types, a representative sample of the different textures must be assembled with which to compare and designate regions. This can be performed by manually defining areas of specific texture types in images similar to those that will be taken during flight operations. As the work presented in this thesis predominantly involves post-processing aerial imagery, a selection of video frames from this flight may be manually classified for training data. This training data is then used to generate samples of each desired texture type, which can be used for data association of ground regions detected during flight. The method for creating these samples is outlined below:

1. Once a selection of typical aerial imagery frames has been assembled, the resulting textures may be manually selected by assigning them a particular colour. For example, red for trees, blue for water, green for grass, grey for roads. These coloured images therefore act as an identifier for the original images.

2. The original sample images are segmented into homogeneous regions using the watershed transform. Methods outlined in section 4.4 are used to limit any over-segmentation. Each detected region is assigned a texture type based on the manually designated identifier image.

3. Each region is characterised for the same parameters that will be used for texture segmentation of the in-flight aerial imagery. The covariances of the parameters for each region are also determined.

The result of this process is a large number of classified texture samples, each defining one of a small set of desired texture types, and each with its own covariance ellipse. The drawback of this approach is that each texture classifier has been developed exclusively to fit a particular set of data, and as such the covariance ellipses will envelop this data. However, it is also desired that each texture set ellipse does not envelop data points from other texture types. For this reason, it is beneficial to use the original training data to refine the texture samples. The process used to perform this is outlined below:
1. The chi-square value of all possible matches between each data point and each texture classifier is determined.

2. This information is reduced to only the best match for each data point to each desired texture type.

3. Any texture classifiers which do not have any data points of the correct texture type associated to them are removed.

4. The sensitivity of these chi-square matches to changes in the mean and covariance of the texture classifier are determined.

5. A linear regression step is used to decrease the chi-square match for each correct association, and increase the chi-square value for each incorrect association, for each texture classifier.

This process can be repeated until some level of convergence is reached. A secondary benefit to this process is that a large proportion of the original texture classifier samples will have been found to be redundant, and will have been removed. This will therefore improve the speed of the texture classifier algorithm when operating in-flight.

The processing of training data results in a set of texture classifiers which can be stored on the vehicle. During operation of the vehicle, any discrete regions detected by the image segmentation process may be compared to these pre-set classifiers. A chi-square similarity metric is employed to compare each region with each texture classifier, with the best matching texture affixed to this region.

### 4.4 Reducing Over-segmentation

A significant drawback of the watershed transform is the tendency to over-segment an image. This will result in the subdividing of homogeneous regions, suggesting that any particular feature is actually comprised of many. This therefore results in the detection of a vast number of detected edges, many of which are not robust, and are not human-identifiable. For this reason, methods must be employed to limit the magnitude of over-segmentation, and to re-combine segments which should not have been divided into new, homogeneous regions.
4.4.1 Gaussian Filtering:

The first method by which over-segmentation may be limited is through a simple filtering technique. A small Gaussian kernel may be employed to eliminate some of the noise in the image. An example of this is shown in figure 4.6, where the watershed transform is applied to an unfiltered image, and also the same image after convolution with a number of small gaussian filter kernels.

![Image](Camera Image Unfiltered Watershed Gauss Kernel = 10 px Gauss Kernel = 5 px Gauss Kernel = 20 px)

Figure 4.6: Comparison of the quantity of watershed over-segmentation with different amounts of Gaussian filtering.

The efficiency in decreasing over-segmentation can be improved by making some observations. Firstly, it is expected that on a small, local scale, edges will mostly act along a constant trajectory. On the other hand, edge traces around noise are much more likely to exhibit high frequency curvature information. For this reason, a large filter kernel should not be necessary. This is compounded by the fact that the watershed transform operates on the gradient magnitude of an image. As determining derivatives such as gradients amplify noise, it follows that reliance on filtering alone to remove noise risks annulling any weak, but useful information in the image.

The drawback of filtering in this manner is small scale features, such as roads, will be removed. Long, thin features such as these may still be detected by considering that they are likely to consist of two parallel gradient responses, which are opposite in direction. Filtering these directly will cause these two responses to cancel out. Instead, by considering the gradient direction, the filter response of these features can be amplified. This is performed by calculating the orientation of the derivatives at each pixel ($\Theta_G$),

95
and re-mapping these gradient response components based on double this angle (i.e., $2\Theta_G$):

$$\Theta_G = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$  \hspace{1cm} (4.7)

$$M = \sqrt{G_x^2 + G_y^2}$$  \hspace{1cm} (4.8)

$$D_x = M \cos(2\Theta_G)$$  \hspace{1cm} (4.9)

$$D_y = M \sin(2\Theta_G)$$  \hspace{1cm} (4.10)

where $G_x$ and $G_y$ are the original image gradients, $D_x$ and $D_y$ are the re-mapped image gradients and $M$ is the image gradient magnitude.

Parallel but opposite gradient edges will therefore become the same sign, and will no longer be filtered out. Conversely, areas where perpendicularly aligned edges are close (such as due to noise) will instead become the opposite sign, and will be filtered out more efficiently than they otherwise would. This double gradient filter is demonstrated in figure 4.7, compared to a standard gaussian filter application. The gradient magnitude response due to noise can be seen to be much lower in the modified gradient case, without the loss of the straight edges.

![Simulated Thin Features With Noise](image1)
![Direct Gradient Magnitude](image2)
![Filtered Gradient Magnitude](image3)

![Coloured Image Gradient](image4)
![Colourised Modified Image Gradient](image5)
![Filtered Modified Gradient Magnitude](image6)

Figure 4.7: Demonstration of gaussian filtering of an image directly (upper right) versus filtering of a double gradient orientation image (lower right). This is compared to the unfiltered gradient magnitude (lower left). Colours describe the orientation of the image gradients.

These methods result in a gradient magnitude solution which captures low frequency, continuous edge features without sacrificing thin features such as roads.
4.4.2 Gradient Threshold:

The filtering of an image is insufficient to eliminate all noise, and as such the application of a watershed transform will still result in significant over-segmentation. A more direct method is therefore required to eliminate the spurious detection of edges from large regions of the image. This can be performed through forcing areas of already low gradient magnitude to zero. This is important as the watershed transform does not consider the actual size of the gradient magnitude in determining boundaries, only which parts are locally maximal.

Figure 4.8: The histogram of the gradient magnitude of an image clearly shows a large number of pixels with minimal gradient. This histogram also shows a long tail of high gradient pixels. This histogram can be used to intelligently choose a gradient threshold.

Small amounts of noise in the gradient magnitude response can be completely eliminated through the use of a limit threshold. Any response below this threshold can be set to zero, ensuring the watershed transform will not consider this area as an edge. The value of this threshold can be heuristically determined based on the properties of each image. As edges in an image can be assumed to be thin boundaries between much larger areas, it follows that the number of edge pixels in an image will be much fewer than the number of non-edge pixels. Therefore, a histogram of the gradient magnitude response can be used to choose this value.

A histogram of the gradient magnitude will result in a determination of the number of pixels exhibiting a particular gradient, separated into predetermined bins. The result of this can be assumed to be a combination of binned edge pixels and binned non-edge pixels. A region between these two binning peaks may then be chosen as the appropriate threshold value. The histogram can be more effectively
calculated by setting a maximum and minimum range on the expected threshold point, and ignoring pixels outside this range. Therefore, the histogram bin with the least number of pixels contained can be assumed to be this threshold point. Figure 4.8 demonstrates the histogram of an example image. Applying the watershed transform to an un-thresholded image can be seen in figure 4.9. Applying the threshold to the image results in a significant decrease in over-segmentation, also shown in figure 4.9.

![Camera Image](image1)
![Unthresholded Gradient Magnitude](image2)
![Unthresholded Watershed](image3)
![Thresholded Gradient Magnitude](image4)
![Thresholded Watershed](image5)

Figure 4.9: The application of a gradient threshold (lower) significantly decreases the instances of over-segmentation compared to the case of not applying such a threshold (upper).

### 4.4.3 Adjacent Region Similarity:

After the watershed transform has been applied to the image, remaining instances of over-segmentation may be reduced through recombination. By analysing the properties of each segment, the differences between adjacent segments can be compared. Should adjacent segments be discovered which exhibit very similar properties, these can be combined into a single, larger segment. This comparison can be performed by examining the same properties which are used during texture classification.

A judgement can be made on the similarity of adjacent segments through the analysis of the chi-square separation of the segment means, relative to the property variance of each segment. This is demonstrated in figure 4.10, where example property distributions are compared.

It is also important to treat the covariances of regions separately during the chi-square calculation. This is because the variance within each segment is a factor which can help determine if segments should be merged or not. For example, a segment of water may have a very small variance, whereas an adjacent segment of trees may have a large variance. The mean properties of the water segment will likely lie
Figure 4.10: Colour content and variation in adjacent regions can be used to identify remaining instances of over-segmentation, and recombine these regions.

within the variance of the tree segment. However, the opposite case may not be true, i.e. the tree property mean may not lie within the water variance bounds. Therefore segments are only merged should both segment means lie within each variance bound. Example results of the segment recombination based on adjacent region similarity process can be seen in figure 4.11.

Figure 4.11: Example of adjacent similarity based region merging, significantly reducing over-segmentation.

The recombination process may be performed iteratively, as each segment merge will result in a change in region property means and variances. This can continue until all segments are determined to be distinct from other adjacent regions. Once this is complete, texture classification can proceed on the remaining regions.
4.4.4 Texture Type Merge:

After texture classification has been performed, the final over-segmentation reduction method may be initiated. By noting the designated textures of each region, adjacent regions of the same type, may be identified and combined into a single segment of this texture. As the image segmentation algorithm, described in this chapter, is designed to identify edge features between different ground texture types, this is a logical step. This can be performed by considering individual binary channels of each particular ground texture type. A pixel bridging algorithm is applied to each of these channels, combining adjacent regions of the same classification. These channels may then be compared with each other, and unclassified pixels adjacent to at least two classified pixels of differing textures, may be assumed to be the true edges. This process is outlined in figure 4.12.

![Camera Image](image)

Figure 4.12: Adjacent regions of the same texture are merged, leaving only borders between differing textures.

4.5 Edge Feature Extraction

Once the image has been segmented, the boundary pixels between different texture types can be extracted as edge lists. These pixel lists may then be filtered to remove high frequency noise, after which the filtered edge may be resampled at a lower frequency. This is done in order to limit the amount of data to fuse, thereby improving the speed of spline fitting. These border points may then be used as sensor measurements. Furthermore, the local in-camera boundary curve slope, and the texture types of the regions defining the border, may also be used as extra information to better describe the edge points.

The first step towards extracting these edge feature points is to identify the edges themselves from the segmented and classified image. This can be performed by identifying the pixels in the image where
the texture classifications change. Once this is performed, it is important to separate any branches, by identifying branch points and deleting these pixels. Upon branch separation, the edge pixel lists themselves may be extracted by determining the end points of the remaining edges. One of these points may be chosen as a start point before removing this pixel, and moving to the next connected edge pixel. This process is continued until the end of the edge is reached, before restarting the process at a new edge end point. Repeating this process until no more end points exist in the image will result in the extraction of all edges. The method of edge extraction is outlined in figure 4.13.

![Segmented Edges Pixel List Filtered Edges](image)

**Figure 4.13:** Edges are extracted by marching along an edge in an image. Extracted edge pixel positions may then be filtered to remove high frequency noise.

Once the edges have been recovered in pixel list form, high frequency noise may be removed through the use of a low pass filter. The next step is to retrieve the gradient of the edge at each point, which can be determined using a simple centred difference method. Once the gradient at each point is determined, test points may be sampled either side of the curve, and used to extract the texture type of the bordering regions. This process is outlined in figure 4.14, after which the slope and texture information may be used to augment the edge information.

![Filtered Pixel List Edge Direction Bordering Texture Samples](image)

**Figure 4.14:** Finite difference methods are used to calculate gradients of the edge points, with perpendicular samples used to extract the bordering texture types.

The final step of the edge extraction process is to reduce the sampling of the data to a more reasonable
level. This can be performed through the pre-determination of a desired angular separation between points, and interpolating new data points along the edge at this spacing. Figure 4.14 also demonstrates this.

4.6 Sun Reflection Mask

A serious visual artefact which can significantly impact on the segmentation of aerial imagery is caused by solar reflections on water. As water will generally appear dark from the air, any solar reflections will exhibit visual characteristics very different from those expected. Therefore these regions are likely to be mis-classified. This is especially problematic considering the goal of the texture segmentation is to allow edge features to be used in a SLAM process. As the position of the sun in the sky can be assumed to be constant for short-term navigation purposes, any viewed solar reflections will appear in the same position relative to the vehicle, no matter its position or velocity. The result of this is that any spurious edges detected due to solar reflections will appear to be stationary relative to the vehicle. Any Kalman updates resulting from innovations relating to these features will therefore reduce the estimated velocity of the vehicle, causing a deterioration of the Kalman estimate. It is therefore important to implement a filter to locate areas of solar reflection, and ignore these regions.

Figure 4.15: Example of solar reflections from a water body.

An example of an image exhibiting a solar reflection from water can be seen in figure 4.15. It is evident from this example that the main characteristic of this effect is over-saturation of the camera, resulting in a large patch of white. Furthermore, waves and ripples on the water surface propagate a
region of high amplitude, high frequency intensity variations around the main reflection. This effect decreases further away from the centre of the reflection, until only dark water texture remains. This high frequency, high variance area is likely to appear similar to tree textures (or other high variation textures), and as such any expected spurious border is likely to exist within this high frequency area. Therefore, the removal of false borders resulting from solar reflections can be performed by detecting these areas, and removing any segmented borders within these regions. These regions can be determined through the use of a simple gradient magnitude filter, such as a Sobel kernel filter. As the region exhibits both very bright, over-saturated pixels, as well as dark water pixels, the resulting gradient response will be very large and easy to detect. Further constraints can be placed on the properties of this region, by considering the colour of the dark water pixels. Should these pixels not be of an appropriate colour for water, it can be assumed that this region is not the result of solar reflections from water.

An example of this solar reflection detector can be seen in figure 4.16, demonstrating that this system is capable of removing spurious edges resulting from these poor lighting conditions.

![Figure 4.16: Example of texture mis-classification due to solar reflections. The use of a sun reflection detector can mask out these spurious borders.](image)
4.7 Results

The imagery analysed in this chapter was recorded by the USYD JabLab aerial test platform (section 1.4). The image sequence was obtained during a flight over Lake Burragorang, Warragamba Dam and surrounding rural areas in NSW, Australia. The vehicle trajectory is shown in figure 4.17, overlaid on Google Earth imagery. This figure also presents an example camera frame captured during the vehicle transit.

![Figure 4.17: Trajectory taken by test vehicle during imagery data collection.](image)

Verification of the texture segmentation algorithm was performed by choosing a number of sample camera frames, and qualitatively assessing the segmentation results. Example segmentations are shown in figures 4.18 and 4.19, showing a number of successful texture segmentations, as well as some examples of the shortcomings of the algorithms used. Examples a, b and c of figure 4.18 show correct segmentations, although they also demonstrate these anomalous areas due to mis-classification of solar reflections as trees.

Examples d and e in figure 4.19 demonstrate another shortcoming of the segmentation algorithm where poor lighting conditions cause severe misclassification of texture. This is especially problematic in the parts of the image where an edge has been resolved in one part of one image, and a different part of the next, which will likely result in large incorrect innovations feeding into the Kalman filter and subsequent solution divergence. This can be mitigated through the map optimisation techniques suggested in section 5.4.5, which will tend to fracture and remove unstable sections of the map. Furthermore, the implementation of a fault detection algorithm, such as the chi-square test outlined in section 5.4.3 will...
help eliminate the worst of these bad associations.

A quantitative analysis of the segmentation accuracy can be made, through the application of the segmentation algorithm, to the training data images. The proportions of resulting correct and incorrect texture classifications, may be quantified and presented in a classification matrix, table 4.1. This shows that the segmentation system is in most part capable of correctly identifying correct textures. An overall count of correctly classified pixels gives this system an 87.5% accuracy. The mis-classifications are mainly predictable, for example significant errors between grass and trees, due to similar colour content. Also, a sizeable proportion of water is classified as trees due to solar reflection effects. Finally, it can be seen that the road texture detection performs particularly poorly. This is due to the edges of these thin features failing to be segmented accurately. Employing an improved road detection algorithm [118] could improve the segmentation of these features.

For the purposes of the work presented in this thesis, robust boundaries between water and trees are
sufficient to demonstrate the functionality of visual airborne spline SLAM. For this reason, classifications of grass and road were removed, and the segmentation classifier re-calibrated. This results in a significantly more robust system, with a combined accuracy of 97.8%. The individual classification breakdowns are shown in table 4.2.

<table>
<thead>
<tr>
<th>Training Class</th>
<th>Training Trees</th>
<th>Training Grass</th>
<th>Training Water</th>
<th>Training Roads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classified Trees</td>
<td>90.84%</td>
<td>14.33%</td>
<td>8.47%</td>
<td>31.33%</td>
</tr>
<tr>
<td>Classified Grass</td>
<td>7.06%</td>
<td>82.58%</td>
<td>0.91%</td>
<td>6.52%</td>
</tr>
<tr>
<td>Classified Water</td>
<td>1.05%</td>
<td>0.01%</td>
<td>87.38%</td>
<td>0.13%</td>
</tr>
<tr>
<td>Classified Roads</td>
<td>1.06%</td>
<td>3.09%</td>
<td>3.25%</td>
<td>62.03%</td>
</tr>
</tbody>
</table>

Table 4.2: Classification matrix for exclusive tree and water reduced case.

The computation speed of the segmentation algorithm is presented per-frame in figure 4.20. This demonstrates that the segmentation execution time generally requires less than 300 ms per frame. This constitutes to an update frequency of around 3.5 Hz, running on an Intel i7-4770K CPU @4.2 GHz. This system also contains 16 GB of DDR3-1600 dual-channel RAM. The Gaussian filtering steps are off-loaded onto an EVGA GTX 780 Classified GPU, due to the high levels of parallel tasks involved. The
algorithms themselves are executed in MATLAB 2013b, with some parts of the processing algorithm written in C++, and compiled as .mex functions. This has been performed to improve the computation speed of the algorithms, however significant speed improvements should still be possible through further optimisation and better use of GPGPU resources.

Figure 4.20 shows that the image segmentation time is predominantly predictable, with minimal variance in calculation time between frames. Periods of significant reductions in computation time, of around 25%, can be seen in some regions. These regions correspond to operations over water, where minimal gradients exist in the imagery, resulting in few, if any detected regions.

Table 4.3 shows a breakdown of the average computation times of different parts of the segmentation and texture classification algorithms. This shows that the algorithm which requires the most processing time is the similarity merge step, which consumes around 40% of the processing time for each frame, and over half of the 200 ms real-time budget.

<table>
<thead>
<tr>
<th>Algorithm Part</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solar Reflection Mask</td>
<td>31</td>
</tr>
<tr>
<td>Gaussian Noise Filter</td>
<td>33</td>
</tr>
<tr>
<td>Gradient Threshold</td>
<td>7</td>
</tr>
<tr>
<td>Watershedding</td>
<td>16</td>
</tr>
<tr>
<td>Similarity Merge</td>
<td>107</td>
</tr>
<tr>
<td>Texture Classification</td>
<td>10</td>
</tr>
<tr>
<td>Texture Merge</td>
<td>9</td>
</tr>
<tr>
<td><strong>Total Mean Time</strong></td>
<td><strong>256</strong></td>
</tr>
</tbody>
</table>

Table 4.3: Average calculation times for particular parts of the texture segmentation algorithm.

The frame computation times shown in figure 4.20 clearly demonstrate that although the real-time goal of 5 Hz has not been achieved, it is certainly within reach. It appears certain that further optimisations
of the algorithms involved in the segmentation process could result in calculation times below 200 ms. However, even if this is not possible, it is clear that two computer systems processing alternating frames could easily produce real-time operations. This is because the segmentation process is temporally independent and as such, does not require any information from previous frames.

4.8 Summary

This chapter outlines the development of a texture segmentation system for the detection of edge curves from aerial imagery. It meets the majority of the desired performance parameters outlined at the beginning of this chapter. These objectives are reviewed below:

1. This system can successfully identify human-recognisable edge boundary features, such as lake and river edges.

2. Texture classifications, with a-priori training data, allow segmented regions to be assigned a human-identifiable ground element type.

3. Noise reduction techniques are employed to remove spurious features without reducing the definition of true features.

4. Methods are employed to improve the robustness of the system to poor lighting conditions, including the removal of under/over-saturated pixels, and masking out areas involving solar reflections.

5. The segmentation system requires an average of around 270 ms of computation time per frame, leading to a refresh rate of approximately 3.7 Hz. This is short of the target goal of 5 Hz for real time implementation, however suggests real-time execution would be possible on more powerful hardware. It is also clear that such a system would still be a reasonable and practical set-up.

The segmentation algorithm outlined in this chapter provides an adequate base for the development of the edge based TAN-SLAM system presented in this thesis. Improvements to both speed and robustness may still be obtained through further refinements to the segmentation algorithm, however it serves as a platform which allows the proof-of-concept of spline based monocular aerial SLAM.
It is important to note that deep learning methods demonstrate improved generality compared to the more directed and specific image segmentation technique outlined in this chapter. This benefit however, is greatly hampered by the increased computational demands and complexity of these algorithms. As discussed in section 2.5.3, even well optimised instances of deep learning algorithms typically require massively parallel hardware, with exemplary performance, to reach real-time execution. Due to the promise they show for accuracy, robustness and adaptability, deep learning techniques will remain an important area of research in aerial visual navigation for the near future. As computational hardware increases in efficiency over time, the implementation of these advanced computational mechanisms become increasingly viable. However, as it stands today, practical applications remain hampered by current hardware constraints.
Chapter 5

Spline-Based Visual SLAM

Visual navigation using SLAM techniques, requires the classification and tracking of viewed features, in order to limit inertial drift. The work presented in this thesis involves the use of spline-based edge features. The result of the texture segmentation process outlined in chapter 4, is a stream of edge points viewed by the camera, as well as gradient information and the textures on either side of each point. This information must therefore be used to define new edge curve features and track how they move through the camera. This chapter outlines the mathematical algorithms required to define new features, and allow them to be initialised into an extended Kalman filter, resulting in SLAM. A simple closest-point association technique is defined to this end, which can also be used to assist in tracking initialised features between frames, by providing Kalman innovations.

This chapter also outlines a number of techniques focussed on improving the robustness of the presented visual spline SLAM system. These include an algebraic method to avoid sensitivity problems arising from spline end-clamping conditions. The use of point-to-spline associations is shown to be beneficial to robustness, as well as decreasing the complexity of the Kalman update process. Furthermore, pre-fusion of new information is demonstrated to improve computational efficiency of the Kalman update process and additionally improving robustness. The consideration of texture and gradient information in the initialisation and association processes, moreover, provides benefits to robustness through the removal of false positive data associations. In addition, a chi-square fault detection technique is employed to further limit bad associations.
The use of freely available terrain profile information is shown to improve the performance of the visual navigation system. Terrain profile information is used to bypass the bearing-only range problem, through direct inference of this parameter, instead of relying on delayed feature initialisations. This range information is obtained by ray-casting techniques through a deterministic, algebraic process.

The work presented in this chapter is separated into a number of sections, which are summarised below:

**Section 5.1** provides an overview of the proposed aerial visual SLAM based navigation system presented in this chapter.

**Section 5.2** presents a ray-casting based method of estimating range to a known digital terrain map. This method is deterministic, predominantly algebraic, and hence is highly efficient.

**Section 5.3** derives algebraic methods of improving the conditioning of splines, relating to loss of parametric sensitivity at the end-points, due to the clamping methods used.

**Section 5.4** outlines the implementation of the data association and Kalman update methodologies used in the development of the SLAM system.

**Section 5.5** presents the results of the proposed navigation system operating on real flight data. This section discusses the accuracy of the system, as well as the computational expense.

**Section 5.6** contains a summary of the outcomes in this chapter.

### 5.1 Proposed Spline SLAM System

This section provides an overview of the proposed spline-SLAM system, outlining how background theory presented in chapter 3 is used to construct this navigation system. This section also acts as an overview, clearly outlining how particular elements discussed later in this chapter are employed in this system.

The simultaneous localisation and mapping algorithm can be separated into four parts:
1. The prediction step consists of inertial measurements used to propagate the vehicle state estimate forward in time. The prediction step of the Kalman filter is defined in section 3.5.1, where a mathematical motion model is used to estimate the future state of the vehicle. This model, and sensitivity matrices, are outlined in section 3.7.1. For the purposes of this thesis, Kalman updates from non-visual sensors are considered to be part of this update step, such as those from static pressure and magnetometer readings. These were outlined towards the end of section 3.7.2, with the method of fusion of these measurements presented in section 3.5.2.

2. The association step involves the identification of which viewed features are known, and which have not previously been seen. This is performed using a closest point association technique to known splines, which was covered in section 3.3.4. This also requires knowledge of the behaviour of the sensor, i.e. a camera sensor model, presented in section 3.7.2. Further information relating to tolerance of branching features is outlined in section 5.4.1, as well as section 5.4.3.

3. Feature initialisation is the main extension of the EKF resulting in SLAM. This involves the characterisation of new sensor measurements into features, which are added to the Kalman state vector for estimation. The SLAM initialisation step is outlined in section 3.6.1, with the spline fitting methodology presented in section 3.3.3.

4. The final SLAM section is that of the Kalman update, the background of which was presented in sections 3.5.2 and 3.6.3. Further information regarding implementation is outlined in section 5.4.2, as well as section 5.4.4.

The visual SLAM technique proposed in this thesis, operates using the re-weighted clamped b-splines outlined in section 5.3.1 to characterise detected edge features. These edges are detected using texture segmentation based on the watershedding algorithm, an overview of which was provided in chapter 4. Point to spline associations are used to determine correlations between splines currently being estimated and new data, which will be outlined in section 5.4.1. Kalman update complexity is decreased using the spline node innovation technique presented in section 5.4.2. The Shuttle Radar Topography Mission (SRTM) digital terrain map is used to estimate the range to detected edge features, as outlined in section 5.2.

With the generation of a new set of detected edge points from the texture segmentation of a new
image frame, the vehicle and feature states can be updated through the following procedure:

1. The range to terrain of all identified edge point measurements is inferred through the use of a known DTEM. If this information is not available, the range uncertainty must be assumed to be extremely large.

2. Associations between each edge point measurement and each spline feature are made. Associations within a reasonable angular tolerance and which pass a chi-square fault detection test, are assumed to be correct associations. Any associations which do not exhibit similar in-frame orientations and texture compatibility are also ignored.

3. Any sufficiently long chain of un-associated edge points undergoes a spline-fitting step, before initialisation into the Kalman filter state.

4. Feature estimate spline nodes near point associations are extracted, and used as an initial estimate for a spline refitting process. The extent of the change in node positions may then be used as a Kalman innovation, with variance derived from the edge point uncertainties fused into the refitting process.

The Kalman filter requires the variance of all sensor measurements to be defined. These edge point measurements are assumed to have a constant uncertainty, relative to their local orientation in the camera frame. It is important to recognise the edge orientation, as any sensor points on the edges are arbitrary, and therefore would be equally valid at any other point on the edge. The existing orientation information from the segmentation process for each point is used to apply this relative variance to the sensor measurements. This is performed by using a sensor variance where bearing and inclination uncertainties are coupled, such that variance is large parallel to the edge, and small perpendicular to the edge. It was determined through experimentation that a perpendicular variance of $5^\circ$ was appropriate, with a parallel variance of $50^\circ$, expressed in degrees of camera FOV coverage.

Should DTEM data not be available, it is advantageous to initialise new features at a prudently chosen height, instead of at sea level. This is because it is expected that viewed features will not be located around sea level, but instead will be distributed around some height above it. The system outlined in this thesis therefore initialises new features at an average of the heights of all features currently being
estimated, when DTEM data is not available. Furthermore, computational complexity of the system can be reduced by removing features from the state vector once they move outside of the camera frame. This can be performed with minimal loss of system accuracy, as once features leave the camera frame, they are unlikely to be significantly affected by any further visual sensor innovations.

5.2 Range Prediction from Terrain

It can be seen from the inverse camera model in section 3.7.2 that in order to predict a reasonably accurate position of a viewed feature, an estimate of the range to the feature is required. Historically this would be performed using a radar system or laser range finder, however due to the reasons outlined in section 2.4.1 this is undesirable. Alternatively, range information can be inferred through using a terrain height map (such as the Shuttle Radar Topography Mission (SRTM) map), and the present pose estimate of the vehicle. This becomes a function of the difference in altitude between the vehicle and the underlying terrain, as well as the vehicle pose. The use of a digital terrain elevation map (DTEM) to estimate range has been demonstrated before [62], however the work presented in this paper does so in a different manner. The previous work [62] uses an iterative approach of gradually extending a ray out from the vehicle, calculating the local vertical, local horizontal (LVLH) world frame position of this projected point as it is extended. This is shown in figure 5.1. The local horizontal $x$, $y$ positions of this point are then used through interpolation of the terrain map, to determine the terrain height directly under this point. Once the ray has been extended to a sufficient length such that the projected point lies below the interpolated terrain height, the iteration step size is reduced. This point is then used to start a re-iteration process to step back along the ray, refining the prediction. Due to the iterative nature of this process, this is potentially highly computationally expensive.

The range prediction method proposed in this thesis, operates by transforming the ground terrain map into the frame of the camera. The range to any desired bearing and inclination coordinate in the frame may then be interpolated based on this ground transformation. This transformation can be performed using the pinhole camera model established in section 3.7.2, and the present Kalman state pose estimate. It therefore removes the need for numerical iteration, improving computation speed. The downside of this approach is that despite the likelihood that a terrain height map will be a regularly spaced rectangular
grid, once transformed into camera frame, this will no longer necessarily be the case. The map will therefore consist of general quadrilaterals, increasing the difficulty and computational complexity of interpolation. Simple steps however, can be utilised to minimise the time taken, which are outlined in figure 5.2. These amount to logical methods of determining which quadrilaterals the target point may or may not lie within. Firstly, the upper and lower limits of bearing and inclination for each quadrilateral are calculated, as in figure 5.2 part a. This can be used to determine a small set of possible intersection candidate quadrilaterals, as the projected measurement ray must lie within these limits. Secondly, by combining the areas of relevant triangles, the true intersecting polygon can be determined. This process, outlined in figure 5.2 part b, can be done by summing the areas of triangles:

\[
\begin{align*}
    s &= \frac{d_1 + d_2 + d_3}{2} \quad (5.1) \\
    A &= \sqrt{s(s - d_1)(s - d_2)(s - d_3)} \quad (5.2)
\end{align*}
\]

where \(d_1, d_2\) and \(d_3\) are the side lengths of the triangle, and \(s\) is half the perimeter of the triangle.

Figure 5.2 part b demonstrates how these triangle sums can be used to determine the true intersecting polygon by evaluating:
Figure 5.2: Bearing and inclination quadrilateral limits can be used to quickly identify polygon intersection candidates (a). The analysis of triangle sums is used to verify which quadrilateral is intersected by the projected measurement ray (b).

\[ A_q = T(P_i, P_j, P_k) + T(P_j, P_k, P_l) \]  \hspace{1cm} (5.3)

\[ A_y = T(P_i, P_j, Y) + T(P_i, P_k, Y) \]
\[ + T(P_j, P_l, Y) + T(P_k, P_l, Y) \]  \hspace{1cm} (5.4)

where function \( T(P_i, P_j, P_k) \) returns the area of the triangle formed by example corner points \( P_i, P_j \) and \( P_k \). \( P_i, P_j, P_k, \) and \( P_l \) are the corners of a test quadrilateral, where \( i, j, k \) and \( l \) belong to integer set \( 1, 2, ..., N \), where \( N \) is the number of datapoints in the DTM. In the example in figure 5.2 part b, these points are \([P_1, P_2, P_4, P_5]\) for the tetragon on the right, and \([P_2, P_3, P_5, P_6]\) on the left. \( Y \) is the sensor measurement, consisting of bearing \( (\chi) \) and inclination \( (\lambda) \). \( A_y \) is then the area of the test quadrilateral. Should \( A_y = A_q \), then the sensor measurement point is inside the test quadrilateral, which is then used to interpolate between the ranges of the four corner points. This interpolation is performed through the mapping of the tetragon corner points to a unit square, which then allows the simple interpolation of the predicted range to terrain. The quadrilateral mapping matrix \( W \), derives the position of the desired point as a function of the quadrilateral corner points. The first row of \( W \) relates to the position of one of the corner co-ordinates. The second and third rows describe the width and height of the tetragon, whereas the final row relates to the measure of rectangularity. The ratios \( L \) and \( M \) are then used to describe the relative position of the target point \( Y \) within this polygon.
\[
W = \begin{bmatrix}
1 & 0 & 0 & 0 \\
-1 & 1 & 0 & 0 \\
-1 & 0 & 0 & 1 \\
1 & -1 & 1 & -1
\end{bmatrix}
\]

(5.5)

\[
\chi = [1 \quad L \quad M \quad LM]W \vec{P}_\chi
\]

(5.6)

\[
\lambda = [1 \quad L \quad M \quad LM]W \vec{P}_\lambda
\]

(5.7)

where \(\vec{P}_\chi\) and \(\vec{P}_\lambda\) are vectors of the quadrilateral corner positions. Equations 5.6 and 5.7 can be solved for the normalised point co-ordinates \(L\) and \(M\), allowing the interpolation of the range prediction:

\[
R_y = R_1(1 - L)(1 - M) + R_2 L(1 - M) + R_3 (1 - L) M + R_4 L M
\]

(5.8)

where \([R_1, \ldots, R_4]\) are the ranges to the quadrilateral corner points.

Should the coordinate lie within multiple quadrilaterals, the solution that returns the smallest range is used.

Clearly, the accuracy of this method is heavily reliant on the vehicle altitude estimate. This ideally would be updated through the fusion of a static pressure measurement, calibrated to the current barometric pressure at the time of any mission deployment. Inferred range measurement accuracy is also affected by any errors in vehicle position or attitude. This can result in navigation solution divergence in some situations, as predicted range bias caused by pose errors can adversely affect vehicle velocity estimates. These errors may in turn compound pose errors. This can be partially rectified by considering the vehicle state precision when inferring range, and accounting for this error through a derived inflation of range sensor uncertainty. Therefore, range estimation will assist in limiting inertial drift while vehicle state precision is high, while resorting to a more typical strict monocular implementation when state precision is low. This maximises the benefits obtained from DTEM based range estimation, while limiting the effects of compounding pose errors, as well as possible navigation solution divergence.
5.3 Spline End Clamping

One of the shortcomings of using b-spline sequences to represent curves is that it is not defined from the first node right through to the very last; instead it is only defined between a weighted average of the first three nodes, to a weighted average of the final three. This is demonstrated in figure 5.3 as an example four node ‘unclamped’ spline. The result of this is that the sensitivity of the ends of the spline, to the positions of the first and last nodes, is very low. This causes sensitivity problems when attempting to estimate the positions of the first and last nodes when fitting a spline to data. This means that the node positions will be extremely sensitive to point associations to the ends of the spline. This shortcoming can be rectified by copying the first and last nodes twice more, such that the average of the first and last three nodes are equal to the first and last node values.

\[
X_{rep} = \begin{bmatrix}
X_1 & X_1 & X_1 & X_2 & X_3 & \cdots & X_{n-1} & X_n & X_n & X_n
\end{bmatrix}^T
\]  

(5.9)

This spline ‘clamping’ results in a spline curve which is fully defined up to the first and last nodes, as shown in figure 5.3.

Figure 5.3: The use of different clamping methods affects the location and distribution of the calculated spline point solutions of parametric variable inputs (e.g. \( t = 0...5 \))
5.3.1 Spline End Re-Weighting

Replicating spline nodes leads to the reduction of sensitivity to changes in the parametric variable, which is a significant drawback. This is easily seen by reviewing the spline gradient at the end points, with spline coefficients simplified to unitary spacing. For $n > 3$:

$$\frac{\partial s}{\partial t} = \begin{cases} 
\frac{k^2}{2} & 0 \leq t < 1 \\
-h^2 + h + \frac{1}{2} & 1 \leq t < 2 \\
1 & 2 \leq t < n - 2 \\
1 - \frac{k^2}{2} & n - 2 \leq t < n - 1 \\
\frac{k^2}{2} - h + \frac{1}{2} & n - 1 \leq t \leq n 
\end{cases} \quad (5.10)$$

whereas for $n = 3$:

$$\frac{\partial s}{\partial t} = \begin{cases} 
\frac{k^2}{2} & 0 \leq t < 1 \\
-h^2 + h + \frac{1}{2} & 1 \leq t < 2 \\
\frac{k^2}{2} - h + \frac{1}{2} & 2 \leq t \leq 3 
\end{cases} \quad (5.11)$$

where $t$ is the real parametric variable defining a point $s$ along a spline of arbitrary length $n$. An example plot of equation 5.10 is shown in figure 5.4, whereas equation 5.11 describes the special case when $n = 3$, which occurs in the case of a two-node spline, i.e. an interval. This interval is therefore the minimum logical case, and as such any situation where $n < 3$, cannot be defined as a curve. This piecewise function can be used to correct the scaling of the parametric variable to force $\partial s/\partial t = 1$. This is performed by inverting the integral of this function to create a map that is shown in figure 5.5. This map defines a new parametric variable $t^*$, which behaves as the ‘optimal’ parametric function described in figure 5.3, between the new limits $t^* \in \mathbb{R}[0, n - 2]$ with $ds/dt^* = 1$. The mapping function now allows the calculation of the ‘clamped’ equivalent variable $t$, where $t \in \mathbb{R}[0, n]$, and which is used to determine the associated spline value.
Figure 5.4: Plot of $\frac{\partial s}{\partial t}$ of a clamped spline demonstrates parametric insensitivity

Figure 5.5: Plot of the mapping function which is used to eliminate parametric end-point insensitivity

5.4 Kalman Innovation Methodology

5.4.1 Point-Spline vs Spline-Spline Association

Spline SLAM methods in the past have focussed on using the difference in location between the predicted positions of previously viewed splines, and those that are fitted to the data points of the most recent observation set, in order to determine whether a feature is known [85]. The issue with this method is that splines are non-unique, resulting in an infinite set of spline nodes which represent the same shape. Therefore any attempts to match splines based on node position will contain a large amount of uncertainty, thereby increasing the likelihood of an incorrect association, and decreasing the accuracy of the Kalman estimate. Furthermore, any branched features that are detected will cause association problems and a loss of accuracy. This may occur because the spline fitting algorithm is likely to fit the resulting splines
inconsistently between frames. This may result in false negative associations unless extra computation time is expended considering matching subsections of splines. An example demonstrating this can be seen in figure 5.6, where uncertainty in texture segmentation has resulted in two different edges being determined for what is essentially the same feature. These splines are clearly not node-compatible, even though half of the detected edge is consistent between frames. It would be possible but reckless to fuse the splines regardless, as this would result in large Kalman innovations of dubious validity, which may degrade the estimation solution. It is also feasible to reject the new spline entirely, however this overlooks the section of spline that is consistent, and is not an optimal use of available data.

Figure 5.6: Demonstration of consistency issues with texture segmentation resulting in increased association difficulty.

The approach taken here ameliorates the association problem by performing data association between measured points and known splines, providing a native tolerance of branching features. It also requires no inter-node or end-point compatibility between freshly viewed edges and known spline features, thus improving robustness in excessively noisy and/or cluttered environments. Associations are made by calculating the closest point on each known spline to each sensor measurement. This step is enhanced through taking edge slope information into account, as well as texture information.

The association method presented by Pedraza [85] operates by fitting a spline to new measurements. This process is outlined in figure 5.7, as the left-side track. Should this new spline be associated with a known spline, this new spline is abandoned. The individual points employed in construction of the new spline are instead used to update the shape of the known spline in the Kalman update step. In comparison, the association presented in this paper (outlined in the right-side track of figure 5.7) does not involve an initial spline fitting step. Instead, a new spline is only fitted to a set of new measurements which are not
associated with previously known splines.

Under certain conditions, this method also has the potential of improved computational times in feature sparse environments, where there is only one reasonable candidate feature for association. In this case, the detected edge points can be immediately fitted to this feature spline. This is in contrast to spline-spline association, where detected points are fitted to a new sensed spline, which is then discarded if it is used to determine a data association. Part a of figure 5.8 follows the example outlined in figure 5.6, showing how sampling points along the viewed edge can allow areas of consistent edge to be identified irrespective of node placement. Furthermore, points corresponding to new branches are not associated, and therefore do not affect the known spline.

![Figure 5.7: Comparison of point-by-point versus refit spline node Kalman fusion ideologies.](image)

### 5.4.2 Point-Spline vs Spline-Spline Updates

This thesis also presents a technique for reducing the Kalman update complexity, by incorporating the Kalman innovations on a spline-spline node position basis, in place of point-spline innovations, as in [85].
Figure 5.8: Points sampled along viewed edges are individually associated with known splines in a. The result of point association is used to initialise or update splines, demonstrated in b.

This is a result of the reduction in data through the spline fitting step, which reduces a large number of point features into a potentially much smaller set of spline nodes. These nodes represent the overall filtered shape of the geometric layout of the point set. This is performed by refitting segments of spline with point associations to these new points, which ensures node compatibility between known and viewed splines (figure 5.7, right-side) allowing a simpler Kalman update step. It also allows unassociated clusters of detected points to be defined as new features, enabling the creation of new branches. An example of a refit node compatible spline, and new splines fit to unassociated points, is given in part b of figure 5.8. This helps to ensure that only reliable sensor measurements are fused into the Kalman state filter, whereas less reliable innovations are rejected.

The result of the re-fitting process is a new spline, which exclusively contains information from new sensor measurements. Furthermore, as this spline was initialised based on the node positions of a known feature, these two splines will be node-compatible. That is to say, the two splines have directly comparable nodes. The result of this is that the difference in position between the re-fitting and original nodes describes the innovation required to transform the curve shape from one state to the other. This innovation can therefore be used to update the Kalman filter, in place of the individual point matches which were used to derive this re-fitted spline. This process is outlined in figure 5.7.

A further benefit of this technique is that because Kalman innovations are performed in the camera frame, reliance on range information of sensor measurements is reduced. Furthermore, as each spline is updated separately, spurious or badly conditioned innovations which would result in a significant
deterioration in the Kalman estimate can be identified before fusion into the filter, and removed. Any remaining re-fitted spline node innovations may then be used to update the Kalman estimate. This helps to improve the robustness of the spline SLAM Kalman update process, by allowing badly conditioned innovations to be more easily identified.

5.4.3 Data Association

The innovations fed into the Kalman filter, are the angular displacements in the camera frame between edge points detected in a new camera image \( Y_s \), and their predicted position through the Kalman filter. This requires the corresponding point on the feature spline of the detected edge point to be determined. For small innovations this can be assumed to be the closest point on the spline to the detected edge point. As there is no algebraic solution to this, it must be calculated numerically. This process was outlined in section 3.3.4.

New edge data points can be compared to the present best feature estimates to determine which sets of points are likely known features, and which are new features. In order to cope with possible branching features, this can be done on a point-by-point basis, determining the closest point on a known spline to each new edge point. Sets of points that match up with known splines can then be used as innovations, whereas sets of points that do not are used to create new features.

Once the closest point on a spline feature to an edge data point has been determined, the probability of association between these points can be determined by considering the uncertainty of the sensor measurement and spline feature:

\[
\Delta Y = Y_s - Y_p
\]
\[
\frac{\partial Y}{\partial X_f} = \frac{\partial Y}{\partial H_p} \frac{\partial H_p}{\partial X_f}
\]
\[
\chi^2 = \Delta Y^T \left( \frac{\partial Y}{\partial X_f} P_f \frac{\partial Y}{\partial X_f}^T + R \right)^{-1} \Delta Y
\]

where \( Y_s \) is the sensor measurement, \( Y_p \) is the predicted closest point on target feature spline, \( P_f \) is the feature prediction uncertainty, and \( R \) is the uncertainty of the sensor measurement. The Jacobian \( \partial H_p/\partial X_f \), is the partial derivative of the spline prediction point with respect to node values, and \( \partial Y/\partial H_p \).
is the camera transformation Jacobian.

The resulting value $\chi^2$ is a measure of the probability that the two points $Y_s$ and $Y_p$ are of the same dataset, and follows a chi–squared distribution. The in-frame gradient of the detected edge can also be used to assist with data association, by applying a limit threshold to the gradient difference of the measurement and associated prediction. This can be further enhanced, with consideration of the textures with which the edge points and splines have been detected and initialised as borders between them. This helps eliminate bad associations, for example providing a clear difference between a hypothetical lake edge, and a tree line located some distance behind it.

5.4.4 Kalman Innovation

Once a set of edge points has been associated with a known feature spline, a sensor measurement spline can be calculated in order to compare to the original known feature spline, acting as a Kalman innovation. The issue with this step is that splines are non-unique, with an infinite set of spline node placements representing the same spline. This causes association issues, as updating the filter with incompatible node placements may cause the filter to diverge. This problem however, can be solved through the use of the known feature spline to seed the spline fitting algorithm. This ensures that the resulting node placement will be compatible with the known feature spline, allowing the Kalman innovation to be made directly using the spline nodes:

$$\Delta I = \left( \frac{\partial Y_p}{\partial X_f} \right)^{-1} \frac{\partial Y_p}{\partial X_f}^T \left[ Y_s - Y_p(Y_s, X_f) \right]$$

The uncertainty of each sensor measurement can then be combined into the uncertainty of the resulting spline, and the measurements fused into the filter, updating the state estimate:

$$\frac{\partial X_f}{\partial Y_p} = \left( \frac{\partial Y_p}{\partial X_f} \right)^{-1}$$

$$R_n = \frac{\partial X_f}{\partial Y_p} R \frac{\partial X_f}{\partial Y_p}^T$$

where $R$ is the uncertainty of the detected edge points, and $R_n$ is the resulting uncertainty of the refitted spline nodes.
5.4.5 Map Optimisation

In order to improve the integrity of the SLAM estimated spline map, and decrease the likelihood of estimate divergence, a number of simple rules can be applied to the spline map. These include combining nodes that are within a certain distance of each other as well as splitting splines at nodes which are sufficiently distant. Further methods include deleting anomalous nodes, such as those which result in a hairpin loop and spline node sequences which are unrealistically short. Furthermore, end nodes of compatible texture type and low angular separation may be joined, as well as extra spline nodes added, in order to extend splines to cover new data points. These processes help to ensure that the spline map remains well conditioned, improving both the robustness of the system and the quality of the spline map.

5.5 Results

The flight data, used for verification of the processes outlined in this chapter, was recorded on a flight over Lake Burragorang, Warragamba Dam, and surrounding rural areas in NSW, Australia. The raw image sequence covers a flight distance of around 48.9 km, with the vehicle flying at varying altitudes of between 300 and 750 m above the WGS84 local sea level definition, and velocities between 43 and 59 ms\(^{-1}\). Image frames were recorded at 5 Hz, with an Ethernet colour video camera with resolution of 1024 by 768 pixels, and horizontal field of view of \((47.5 \pm 0.5)^\circ\). Synchronisation of image frames, IMU data and magnetometer measurements is performed using time stamps on the data as to when it was collected. Verification of the spline SLAM algorithm, and the results presented in this paper, is performed on a 400 second section of this flight path. This path section contains sections of robust edge features (forest/water boundaries), as well as periods where no features are viewed. This corresponds to a flight path of just over 21 km. The recorded visual and sensor data were then post-processed to determine the resulting accuracy of the vehicle pose estimate.

5.5.1 Statistical Analysis

One of the fundamental underlying principles of the Kalman filter (and hence SLAM) is that all noise is Gaussian (normally distributed). Figure 5.9 outlines the distribution of the Kalman innovations obtained
by forcing the vehicle position to follow the GPS/INS truth solution. This shows that the innovation distributions are not strictly Gaussian, with a sharper peak and wider base. The range distribution is marginally asymmetrical, however range and bearing innovations are mostly symmetrical. This suggests that the extended Kalman filter may not be the most accurate estimator for this application, however as the tail distributions still decay quickly, inaccuracies should be limited. Furthermore, higher order modelling techniques used in alternative filters such as the UKF or particle filter, will severely restrict real-time implementations.

Figure 5.9: Visual Kaman innovation histograms compared to Gaussian distribution of equivalent mean and standard deviation for range, bearing and inclination visual sensor measurements.

5.5.2 Navigation Solution Accuracy

A comparison of the trajectory estimates of the dead-reckoned and spline SLAM solutions can be seen in figure 5.10, compared with the GPS/INS truth. This figure shows both the trajectory with and without the use of DTEM data, to estimate the range to viewed features. It is clear from this figure that the visual SLAM process decreases the integration drift of the IMU, keeping the estimated trajectory much closer to the true flight path than dead-reckoning alone. It is evident that this significant navigation accuracy improvement is obtained over dead-reckoning for both the DTEM range estimation case, as well as when the DTEM is not utilised.

Figure 5.11 shows the predicted spline map generated in the SLAM process, overlaid on Google Earth imagery. This image shows that the detected features are mapped close to their true position, and in most cases contain recognisable shape information which can be visually associated with the Google Earth imagery. It also shows how errors in the vehicle localisation estimate results in errors in the mapped
Figure 5.10: Comparison of the trajectory estimates of dead-reckoning, versus SLAM both with and without DTEM range estimation.

Figure 5.11: The SLAM trajectory with range estimation, superimposed on Google Earth imagery. Also shown is the resulting spline map from the SLAM process.

The root sum square position error of the vehicle estimates can be seen in figure 5.12. It is clear that the SLAM position drifts much more slowly than dead-reckoning alone, mostly staying within 150 m of the true position. It is clear that although the SLAM systems with and without range estimation perform differently, they exhibit similar levels of position error.

The estimated position of the vehicle can be seen to remain relatively constrained during periods where characterisable features are detected, and exhibits quadratic dead-reckoning drift, when features are not visible. It is interesting to note that once spline SLAM updates are resumed after periods of dead-reckoning, cross-coupling between position and velocity uncertainties during inertial integration...
Figure 5.12: Absolute position error for 400s flight for dead-reckoning, versus SLAM both with and without DTEM range estimation. The Kalman update complexity variation with time is also shown.

can result in a reduction of localisation error. This process also results in a small decrease in variance. This is due to the resumption of innovations constraining velocity, and subsequently quickly correcting a portion of the accumulated position error from velocity drift. This phenomenon is especially pronounced in the first few minutes of flight, whereas the second half of the flight does not show clear reductions in position error. It does however, still demonstrate clearly the successful limiting of dead reckoning drift.

The reasons for the differences between performance characteristics in the first and second halves of the flight can be explained by noting the velocity estimate comparisons in figure 5.13, as well as the terrain the vehicle is traversing, which can be viewed in figure 5.11. It is clear that the distinct edge features present during the first half of the flight help to constrain the vehicle velocity drift, resulting in small perturbations about zero error. The second half of the flight exhibits a long, predominantly straight, edge. This results in difficulty obtaining clear feature tracking parallel to this edge. Consequently, there are reasonably constant velocity biases evident in the latter halves of the velocity plots.
Figure 5.13: Comparison of forward and sideways velocity components

Figure 5.13 shows the benefits of using the SLAM algorithm to constrain velocity drift, as it can be seen that the forward velocity estimate varies less than that of basic dead-reckoning. The range estimation can be seen to have a minor beneficial effect on the forward velocity estimates, however it is clear that the SLAM process itself successfully limits inertial velocity drift. This is despite the lack of robust bounding of velocity in either case, either due to lack of range information, or range inference errors due to position drift.

The sideways velocity component of the aircraft estimates in figure 5.13 demonstrates a more robust benefit of visual SLAM, as the movement of features through the camera can be reliably used to accurately estimate the sideways movement of the vehicle. It can be seen that the lateral motion of the SLAM estimates are bounded, and therefore vastly superior to the dead-reckoning estimate for most of the flight. When combined with the attitude constraints granted by the fusion of the magnetometer, this results in strong trajectory constraints on the SLAM estimate. The fusion of a magnetometer alone is not sufficient to provide this constraint to dead-reckoning, as seen in Figure 5.10, resulting in significant lateral drift. The reasons behind this divergence are discussed in section 2.3.1.

The spline map generated as part of the SLAM process is shown in figure 5.11, for the case of DTEM range estimation. An example comparison of the map quality between the case where this range information is used, and the case where it is not, is shown in figure 5.14. This small part of the resulting map clearly shows that the use of range information greatly improves the robustness of the mapped features. The increased range uncertainty in the case where DTEM data is not used produces much more altitude noise in feature position estimates. This increase in altitude noise culminates in lateral displacements due to viewing angles, greatly impacting the reliability and accuracy of the SLAM map.
Figure 5.14: Comparison of mapped edges when DTEM data is used to estimate range (black), versus when it is not (red).

5.5.3 SLAM Computation Times

One of the key indicators of a functional aerial navigation system is update frequency, as any such system would need to work in real-time. For this reason, the per-frame calculation times have been examined. As stated in chapter 4, one of the goals of this thesis is to process new features from imagery frames at a rate greater than 5 Hz, in order to provide real-time execution. Figures 5.15 and 5.16 demonstrate that the outlined navigation system operates at a rate above 5 Hz both with or without DTEM range information. It is clear from these figures that the range estimation has only a minimal effect on the required processing time. It is important to note that these times do not include processing time required for image segmentation, as this is assumed to occur separately, on different hardware than the SLAM filter. Nevertheless, the same computational hardware has been utilised for analysis purposes of the SLAM computation times, i.e. a 4.2 GHz quad-core, i7-4770K CPU.

Figure 5.15: Frame execution times for SLAM algorithm, without range estimation.
Figure 5.16: Frame execution times for SLAM algorithm, with range estimation.

A more detailed computation time breakdown of the systems are presented in figure 5.17. This demonstrates the average computation time per frame spent performing different actions. It is clear that a substantial amount of time is spent processing measurements, involving the extraction of edge points from pre-segmented images, including textures and orientations. It is also clear that the Kalman update step is also rather intensive. It is however evident that the average frame processing time is around 50 ms, which is far less than the maximum of 200 ms required for real-time operation.

Although the mean computation time per frame is useful, it is more important to consider the processing time of the worst-case frames. Figure 5.18 outlines the time breakdown of the frames which required the most costly calculations to complete. These show that the data collection step, involving the processing of the static pressure and magnetometer sensors, remains relatively constant. However, all other areas require more computation time. The most pronounced change is predictably the Kalman update step, as this step scales poorly with the number of estimated feature states. The edge point processing step will increase linearly with the number of edges detected in a frame. It is also clear that the Kalman prediction step requires more time to complete with more feature states.

The required computation time of simulations is not noticeably affected by the use of estimated range information. This is clear from both figures 5.15 and 5.16, where the computation time traces are almost identical, albeit with minor variations. Figures 5.17 and 5.18 also demonstrate this independence. It is therefore clear, that the only considerations necessary in determining if DTEM based range estimation should be employed, is navigation solution accuracy.
5.6 Summary

This chapter outlines the development of a SLAM based visual aerial navigation system which operates using human identifiable edge features. This system successfully limits inertial drift of a vehicle, resulting in position errors consistently bounded to just 150-250 metres, over 400 seconds of operation under GNSS-denied conditions. This is in stark contrast to inertial navigation drift, which results in over 4 kilometres of position error over the same time period. This system also demonstrates a capability for real-time operation, with all visual data fusions in this test sequence, occurring well within the 200 ms time envelope required to process each frame of the 5 Hz imagery stream.

Digital terrain data can be used to estimate the range to viewed features, without having a noticeable
impact on computation times. This SLAM system is also capable of producing an accurate map of terrain edge features, which is also human-recognisable. Methods outlined to improve the robustness of the visual update system and provide tolerance to any potentially branching edge features, help to ensure the accuracy of this map. These methods also ensure that any uncertainties in the exact edge location of viewed features, do not cause any significant deteriorations in the Kalman vehicle state estimates.

It is interesting to note that the use of feature range estimates does not noticeably improve the accuracy of the system. This could be considered unexpected, as using extra information in the case of range estimates should improve the solution. However, as the accuracy of the inferred range measurements depends on the pose estimate of the vehicle, these range measurements will be biased in some cases, potentially diverging the solution. Furthermore, the use of range estimates makes the system highly sensitive to altitude or DTEM errors. If this information is not used, the navigation system may be much more tolerant to these errors. This is because the use of a DTEM provides some level of absolute information, if this data is not used, innovations are completely relative. Any errors in altitude will therefore correspond to a subsequent mapped feature displacement, preventing any biases in inferred velocity. Errors in the DTEM would clearly be irrelevant, as it is not used.

Although range estimation does not significantly improve navigation accuracy, the benefits of using the DTEM are clear from figure 5.14. This figure clearly shows the quality of the mapped edges with the use of range estimates is vastly superior to the quality of those made without. This will help with spline based data associations used for TAN, as the ability to match splines will be highly dependent on minimising noise or systematic errors. A second benefit of the use of the DTEM is that bearing-only SLAM cannot constrain the velocity of the vehicle. This can be seen conceptually by considering that the range to features must be inferred from the velocity of the vehicle. These ranges, in turn, are used to estimate the vehicle velocity. This cyclic process is therefore susceptible to drift without any additional external information. The addition of range measurements from the DTEM information breaks this cycle, preventing velocity drift. This can only occur in the situation where the vehicle pose is known to a high degree of accuracy. There is a clear second benefit then, in the use of DTEM range estimation in the case of TAN-SLAM, despite minimal, if any, benefit to navigation with unassisted SLAM.
Chapter 6

Terrain Feature Assisted Navigation

The SLAM algorithm is useful for limiting inertial integration drift, however it cannot by itself eliminate drift completely. To properly bound the position estimates of a vehicle, absolute position information is required, as opposed to the use of exclusively relative measurements as with traditional SLAM. This thesis therefore presents the idea of using a known curve database as a terrain feature map. This map may be used during flight to localise the vehicle through data associations with currently viewed features.

This process can be separated into three distinct sections:

1. This terrain aided SLAM system relies on the creation of an edge feature terrain map. This map can be derived from sources of geo-referenced aerial imagery, such as Google Earth, or any other such source. In order to improve computational efficiency of the association step, this aerial imagery can be pre-processed into edge curves, which can be stored using a spline representation. This also has the benefit of significantly decreasing the amount of data storage required to contain this map. This process is outlined in section 6.1.

2. During flight, viewed edge curve features may be associated with this terrain feature database, resulting in vehicle state absolute position innovations. This data association methodology is based on comparisons of gradient, curve perimeter length and local curvature. Reliance on these high level parameters provides tolerance to the non-uniqueness of spline nodes. Section 6.2 describes this association process.
3. Once a data association has been made between currently viewed curves and known edge features from the map, this innovation is used to update the Kalman filter state estimates, for both the vehicle pose and all viewed features. Fault detection techniques are employed to prevent the fusion of any spurious associations, improving the reliability of the system. This process therefore bounds the position estimate of the vehicle, negating accumulated inertial drift. The precise methods of fusion of these innovations are outlined in section 6.3.

An important parameter, related to the computation time required to perform these spline feature data associations, is the search space used. This search space governs the extent of the local database spline map which is analysed for potential associations. It can also be used to drop spline nodes from the Kalman state vector once they move sufficiently far outside of the camera frame field of view. This parameter is defined, and its usage discussed in section 6.4.

Potential systematic errors can result in a deterioration of TAN-SLAM performance, should the feature database be incorrect for any reason. One such reason this can occur is through changing water levels of water bodies. Due to the surrounding terrain profile, water level changes can result in large water edge position movements, which in turn will bias associations made to these features. For this reason, a method for counteracting these errors is developed in section 6.5, should the correct present water level be known. In the case where this is unknown or unavailable, discrepancies in the expected edge locations can be used to estimate the current water levels, as in section 6.6.

This chapter contains the following sections:

Section 6.1 outlines the procedure for building the spline based feature map from geo-referenced aerial imagery sources.

Section 6.2 presents a gradient and curvature integral based spline association method capable of matching open, partial curve features. This method also exhibits tolerance to partial occlusions, as well as inclusions.

Section 6.3 describes how curve association innovations are fused into the Kalman filter to provide TAN updates to the SLAM filter. This section also shows how the search space of the feature map can be limited based on previous curve matches, improving computational efficiency.
Section 6.4 defines a deterministic method of removing feature state nodes from the SLAM filter, once they move a sufficient distance outside of the camera frame. This method may also be used to limit the extent of feature splines which are used in the curve association process.

Section 6.5 outlines how a-priori knowledge of changes in water levels can be used to reduce position biases caused by these changes.

Section 6.6 describes how differences in the shape and expected locations of edge features, can be used to estimate changes in the water levels of viewed water bodies.

Section 6.7 verifies the outlined spline feature based TAN-SLAM algorithm using real flight data. The accuracy of this terrain assisted extension to SLAM is compared to the unassisted SLAM system developed in chapter 5. Computation time is also evaluated and discussed.

Section 6.8 provides a summary of the outcomes presented in this chapter.

6.1 Feature Database Creation

6.1.1 Google Earth Imagery

Google Earth provides a convenient database of terrain information, for most of Earth’s landmass. This predominantly comes in the form of aerial imagery, from a combination of specialised satellites and aircraft. The consequence of this is that the information contained in Google Earth is highly human-recognisable, as it is visual information. Furthermore, as it is predominantly aerial photography, it will generally be of the same form as the sensor measurements available to the proposed system, ie. aerial digital imagery. This, combined with the ease of collection from, and extensiveness of information contained within the service, makes it a highly useful tool.
6.1.2 Texture Segmentation

A drawback of the information available from aerial photography, both from Google Earth and from
digital video, is the complexity, and magnitude of the data. Storing large areas of digital imagery becomes
logistically difficult. Furthermore, as digital imagery is effectively a large matrix of pixel elements,
data association of digital imagery is highly inefficient, as it contains a large amount of inconsequential
information. The goal is therefore, to process this imagery in such a way as to greatly compress the
available data into a small amount of highly representative information.

The method used to perform this data compression in this work is texture segmentation. This is an
analysis technique which identifies areas of different texture in an image, and designates each separate
area to a representative type. These texture types are water and forest in this work, however they may
also include road or tarmac, concrete, sand, grass, etc. Borders between adjacent areas of different type
may then be identified, and characterised as an edge curve. The textures defining this boundary can also
be recorded as further information to help define a unique feature.

6.1.3 Imagery Inconsistency

A significant limitation of Google Earth imagery is uncertainties in the absolute positions of the imagery.
Errors in satellite and aircraft position or attitude, as well as some perspective or viewing angle effects,
will cause inaccuracies in the overlaying of imagery onto the Google Earth maps. This results in an
error between the apparent latitude and longitude of a particular ground feature in Google Earth to
its true position. This can be quantitatively analysed by recording historical imagery, and comparing
the apparent latitude and longitude of a clear landmark, such as the centrepoint of a roundabout road
intersection. Figure 6.1 demonstrates this, with the black dot in each frame representing the average
reported latitude–longitude position by Google Earth of the white centre of the roundabout from each
image. The roundabout itself is located at Warragamba Dam in NSW.

The result of this is a discernible, systemic bias of features in the map, which may vary in magnitude
and direction across the terrain. Figure 6.2 quantifies these errors, and outlays their distribution around the
mean. This bias error is problematic, as one of the fundamental assumptions upon which the Kalman filter
is based on, is that all noise is Gaussian, with zero mean. Errors in the projection of imagery into Earth
Figure 6.1: A roundabout at Warragamba Dam is used as an example of position bias in Google Earth imagery.

Latitude and longitude coordinates, however, are likely to be consistent over the entire image projection. Therefore, errors will clearly not have zero mean. Instead, these errors will exhibit a semi-constant bias on a small to moderate scale, depending on the aerial image sizes, and the evolution of the state error of the aerial photography vehicle. The end result of these systemic errors is that position innovation updates will cause the vehicle state estimate to converge to a small bias value. Adding to this problem, repeated measurements may result in the certainty of this position estimate to fall far below that of the error. This issue can be moderated by taking the imagery uncertainty into account in the development of the association variance. Furthermore, the presented work is aimed towards successful localisation and navigation of a vehicle traversing dozens or even hundreds of kilometres. Therefore, it is important to realise that an unavoidable lower bound on position uncertainty of 5-10 metres, is effectively irrelevant.

Figure 6.2: Distribution of position biases for 14 sets of Google Earth imagery and confidence bounds (5m for $1\sigma$ and 10m for $2\sigma$).
6.2 Edge Data Association

Terrain aiding of the SLAM filter involves fusion of absolute position information into the Kalman filter, requiring a sensor measurement of world position. This can be done by the association of the SLAM edge map estimation, with a pre-processed edge map, such as that retrieved from Google Earth imagery.

6.2.1 Gaussian Filtering

A significant drawback to data association based on curvature is the amplification of noise due to differentiation of the edge shape. This can result in the characteristic shape information being suppressed by the characteristics of even a comparatively small amount of noise. This drawback can be limited through the use of a Gaussian filter, however care must be taken to avoid filtering out useful information. For this reason, the typical spatial frequencies of the noise must be determined, as must the frequencies of the descriptive shape information. A Gaussian filter may then be applied, with a wavelength above that of likely noise, but significantly below any likely shape information. This sets a hard floor on the curvature content to be analysed.

6.2.2 Adaptive Filtering

Although low frequency information in terrain edge features is what makes them unique, some examples of edge shape may be best characterised by higher frequency information, as they contain very little variation at low frequency. For this reason, the frequency information contained in the edge itself must be used to assign an appropriate low-pass filter frequency to eliminate noise, without degrading the relevant shape information of the edge.

The adaptive spline filtering process employed in this paper resamples nodes based on the cumulative angular change along the spline. This is done by initially identifying the inflection points along the spline, which can be performed through the algebraic calculation of zero curvature points. Inflection points may
be determined by reviewing the spline weighting equation:

Let $\Xi X_i = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} X_F \begin{bmatrix} i \\ \vdots \\ i+3 \end{bmatrix}$ \hspace{1cm} (6.1)

$s = \begin{bmatrix} h^3 \\ h^2 \\ h \\ 1 \end{bmatrix} \Xi X_i$ \hspace{1cm} (6.2)

$s' = \frac{\partial s}{\partial t} = \begin{bmatrix} 3h^2 & 2h & 1 & 0 \end{bmatrix} \Xi X_i$ \hspace{1cm} (6.3)

$s'' = \frac{\partial^2 s}{\partial t^2} = \begin{bmatrix} 6h & 2 & 0 & 0 \end{bmatrix} \Xi X_i$ \hspace{1cm} (6.4)

where $h$ is the parametric variable governing the distance of a point $s$ along the local spline segment. $X_F$ is the feature state vector, and $t$ is an extended parametric variable governing the distance of a point $s$ along an entire spline sequence.

These derivatives can now be used to determine the two-dimensional curvature of points on a spline (see appendix 10.1):

$$C = \frac{s'_x s''_y - s'_y s''_x}{\left[(s'_x)^2 + (s'_y)^2\right]^{3/2}}$$ \hspace{1cm} (6.5)

where $s'_x$ and $s'_y$ are the derivatives $\partial s/\partial t$ with respect to node dimensions $x$ and $y$. $s''_x$ and $s''_y$ similarly represent the second derivatives. This curvature equation may then be solved analytically for $h$ values where $C = 0$ for each local set of four spline nodes, resolving the inflection points $h_{inf}$. Should $h_{inf} \in \mathbb{R}[0, 1]$ then a local inflection point exists.

Once inflection points have been calculated, the cumulative angle change between successive inflections is found. By comparing this to a desired baseline ($G_0$), a reasonable number of nodes may be sampled along this segment. This baseline may be chosen dynamically by analysing the existing gradient changes between inflections. The result of this process is that areas of low curvature will be sampled with a low node frequency, whereas regions exhibiting high curvature will still contain sufficient node density.
to retain a good approximation of their true shape.

\[ \Delta G_{ab} = \int_{t_{inf}(a)}^{t_{inf}(b)} C \, dt \]  
\[ N = \left\lfloor \frac{\Delta G_{ab}}{G_0} \right\rfloor \]  
\[ \Delta t = \frac{[t_{inf}(b) - t_{inf}(a)]}{N} \]  
\[ t_{nodes} = t_{inf}(a) + \left\lceil (1, 2, \ldots, N) - \frac{1}{2} \right\rceil \Delta t \]

where \( \lfloor x \rfloor = \text{round}(x) \), and \( t_{inf}(a) \) and \( t_{inf}(b) \) are two adjacent inflection points. \( \Delta G_{ab} \) is the cumulative change in gradient between these inflection points, and \( N \) is the number of resampled nodes.

Once the new node separations are calculated, point innovation based spline refinement is used to ensure the new spline fits the shape of the old spline, as well as possible. A secondary benefit to this process is that spline nodes are sampled at a much lower frequency, therefore decreasing the amount of data required to store the curve shape. An example of this adaptive filtering can be seen in figure 6.3, showing the redistribution of nodes to best capture areas of complex shape, while minimising node placement in areas of low variation.
6.2.3 Shape Wavelength

A simple way of calculating the position offset between two shifted, but otherwise identical datasets, is to perform a closest point match innovation regression. This involves sampling points along one dataset, and determining the closest points in the other set, to each of these samples. These innovations can be used to resolve an overall shift and rotation to improve the correlation between the two maps. This method is effective for characterising small errors, however becomes less reliable for larger offsets as multiple contenders can lead to false positive associations. The point at which this happens can be qualitatively resolved as half the dominant wavelength of the edge frequency information. This can be seen by reviewing figure 6.4, demonstrating that closest point match refinement methods can only recover comparatively small quantities of drift error.

![Figure 6.4: Point matching techniques are only valid for small errors](image)

6.2.4 Tokenisation

A convenient method for performing data association which is less reliant on position, is to reduce a complex set of information into a small number of data sets, which ideally, uniquely and robustly describe the data. These individual data sets can be thought of as tokens, and can be used to associate complex features by simply comparing a small set of numbers. These elements are outlined in figure 6.5. This greatly simplifies the problem, and decreases the computation time of data association.
Due to the fusion of a magnetometer into the state estimate, it can be assumed that the vehicle heading is reasonably accurate. For this reason, the components \((d_x, d_y)\) of the interval vector \((d)\) between successive inflection points in navigation frame can be used to match tokens. This can be enhanced by the use of the integral of the curvature over this segment, which is the angular change in curve direction \((h_1 - h_0 = \Delta h)\). Finally, the ratio of curve length \(|s|\) to the straight line distance \(|d|\) between inflection points, gives a further simple indication of the shape of the segment. The resulting token \(T\) can therefore be expressed:

\[
T = \begin{bmatrix}
  d_x & d_y & \Delta h & \frac{|s|}{|d|}
\end{bmatrix}
\tag{6.10}
\]

Figure 6.5: Token data including curved to straight distance ratio \((s/d)\) and angular change \((h_1 - h_0)\)

6.2.5 Over–Representation

Tokenisation of edge data results in a risk of false rejection of data associations, due to incompatible representations of similar data. It may occur in this case due to the reliance on inflection points for token calculation. Should an edge have a weak inflection point (figure 6.6), it may be detected in one dataset, but not in the other. This will result in differing token representations of the edge, preventing the feature that the token represents from being matched. A solution to this problem can be implemented by searching for these weak inflection points, and defining characterising tokens for both the case of the
inflection point being present, and not. Potentially weak inflection sets may be determined by applying a threshold to the angular change of each token. Any tokens with a minimal cumulative curvature may then be combined with the two adjacent tokens resulting in a new, over-representation token. This allows data association to be correctly performed even when particular inflection points are only detected in the known data set and not the viewed edges, or vice versa.

![Figure 6.6: Small changes in shape can cause non-robust tokens](image)

6.2.6 Candidate Recombination

Comparisons of tokens may result in a large number of potential associations, due to any similarities between tokens. This is likely in any areas exhibiting periodic features. Examples where this would be common include grid-like urban road networks, or stretches of river where edge boundaries contain reasonably uniform frequency content. These periodic features will exhibit similar shape information, and as such, each permutation of these alike features will present a possible match. However, as this method involves breaking up curves into small sections and matching each permutation of the two datasets, many of these associations will describe the same overall position innovation. For example, it would be expected that correct matches between different parts of the curve would result in similar innovations. Therefore, the resulting innovations of each potential match may be compared, and similar innovations combined. Furthermore, multiple similar innovations are most likely indicative of a correct association. Consequently, these remaining combined innovations can be ranked based on the number of innovations that were combined.
6.2.7 Critical Points

Complex curve shapes may also be analysed for areas of interest, thereby providing a second method for comparing continuous features using a discrete method. These points of interest, or critical points, are considered to be local points of either maximum curvature, or rate of change of curvature. These provide human-identifiable points, as they represent sudden changes of direction of the curve, or s-bends. The critical points therefore represent a flag, designating a particular point on the curve that may be uniquely recognised. An example of critical points is provided in figure 6.7.

![Critical Points Diagram](image-url)

**Figure 6.7: Areas of local maximum curvature and maximum curvature change**

6.2.8 Critical Region Comparison

Once a series of critical points has been determined, and a potential match has been made between tokens, the likelihood of a correct match is evaluated by comparing regions around critical points. Should the position shift resolved by the token match be applied, critical points of a given type may be matched based on proximity. Any critical points within a certain distance from each other constitute a potential verification of the token match.

The quality of the critical point match is assessed by sampling points around the critical marker from the viewed curve set. These points may then be compared to similarly sampled points around the associated critical marker from the known curve database. Differences in the shape may be quantified.
by determining the average point separation in the comparison. The robustness of this step is greatly increased by the application of a sliding window method, moving the sample area over a small section (figure 6.8). This helps decrease the effects of noise influencing the specific location of the critical points in each set. The smallest cost returned from window position therefore, may be used to verify the critical point match, and in turn, the overall token match. Should this minimum be below a certain threshold, this area match may be verified, thereby potentially validating the original token match.

![Figure 6.8: Critical points on a spline may be directly compared](image)

### 6.2.9 Point Match Regression

The drawback of performing data association on gradient and curvature based tokens, is that due to filtering and noise, the position difference of a token match is likely to contain a large amount of variance and error. The matching process error will however be less than the wavelengths of the edge shape information, as the size of each token will likely be around half the local dominant wavelength. This allows direct point-innovation regression to be used to greatly improve the association solution. This step is repeated until the change in position from each iteration becomes small enough to signify convergence.
6.2.10 Choose Best Match

The comparison of tokens will likely result in a number of possible matches, and depending on the shape of the features, more than one may be verified by the critical region comparison step. It therefore becomes necessary to choose the best match candidate from the possible contenders. This can be performed by considering the remainder innovations from the point match regression, for each association. The smallest mean innovation may be assumed to be the best match, and should this mean fall below a certain threshold, it may be considered as a correct association, and subsequently fused.

6.3 Kalman Filter Sensor Fusion

Once an absolute map match has been made between estimated and known features, this information can be used to update the Kalman estimates of the vehicle, and all feature states. This can be done by using the position shift implied by the map match as a sensor innovation, and the uncertainties involved in the spline matching algorithm as the sensor variance.

6.3.1 Map Match Fusion

The result of the edge data association is the position shift, which is required to better align the known and viewed estimate feature maps. This shift represents the sensor innovation to be fused into the extended Kalman filter. Also required is a measure of the position innovation uncertainty of the matched object. This uncertainty can be used as the sensor variance required for fusion. As outlined earlier in section 6.2.9, the matching process involves data association of a number of sampled points along the viewed edges. This helps in the numerical development of the match uncertainty, as each match has a local innovation, and a local gradient.

An obvious contributor to the uncertainty of the map match is the distribution of point based residual innovations resulting from the point match regression (section 6.2.9). This is directly related to the quality of the map match, as large variations in these innovations suggest the two maps are less consistent. This uncertainty can be easily characterised by calculating the covariance matrix, or spread, of the innovations as individual sensor measurements.
A second, more vital source of uncertainty comes from the edge shape itself. Consider the situation of matching a long, straight road. The straightness of the road prevents any meaningful position information from being recovered in a direction parallel to the road. However, clear positional information will constrain the vehicle location perpendicular to the road. Effectively what this means, is that the association will result in the localisation of the vehicle to be somewhere over the road, but unsure how far along the road. This concept can be applied to the filter by considering the gradient of the viewed edges, and using this information to derive an authentic match variance. By analysing the gradient at each regressed closest-point match, individual point variances may be defined to be relatively uncertain parallel to the local curve, and relatively certain perpendicular to this local curve. These individual point variances may then be combined, generating an uncertainty of the match based on the curve shape. This process is described further in section 7.5.2.

6.3.2 Search Space Prediction

An issue that arises from terrain feature aided SLAM is that as absolute position data is fused, the variance of the vehicle position no longer has a bounded minimum. In the case of unassisted SLAM, new features are initialised with an uncertainty derived from, and coupled with, the vehicle uncertainty. This map is then used to localise the vehicle, with the map variance and cross-coupling, limiting the level of convergence of the vehicle uncertainty bounds. This means each innovation affects both the feature and vehicle pose estimates. The cross-coupling between the vehicle, and SLAM feature positions, prevents the variance from reducing to a level lower than the initial vehicle uncertainty when these features were first viewed. However, when associating with known database features, absolute position data is fused into the vehicle estimate. These fusions utilise innovations which no longer affect feature positions, as the true locations are already known. These innovations therefore allow the vehicle position variance to decrease below this level. Furthermore, cross-coupling in the Kalman state covariance matrix also results in a reduction in any feature uncertainties.

Generally, a reduction in the variance of state estimates is beneficial, as it suggests an increase in certainty, and accuracy of the estimate. This occurs because one of the fundamental assumptions made in the application of the Kalman filter, is that all noise is Gaussian, with zero mean. Due to the nature of the creation of the known feature database, this assumption cannot be relied upon. Figure 6.1 gives an
example in which the noise may not have zero mean. Furthermore, errors in the texture segmentation process are likely to result in more biased errors.

The result of this is a high degree of certainty in vehicle position, as well as map biases which may be far greater than any reasonable uncertainty bound of the vehicle position. This risks correct position innovations being rejected through likelihood based fault detection, preventing the state estimate from converging to a better localisation solution. The proposed solution is to estimate a search space offset based on past map innovations. This provides more freedom to the filter, allowing map matches to be identified more reliably, and with a smaller search area. This has the added benefit of limiting the search space, improving computation time, and lowering the possibility of incorrect associations.

Should the vehicle operate for an extended period without terrain feature updates, the pose estimate will degrade due to dead reckoning drift. The corresponding state variance expansion will also result in an increase of the required search space within which to consider data associations. Upon resumption of absolute position information, the filter will converge exponentially to a new value close to the true position, considering vehicle attitude errors and biases. This convergence will occur at a rate governed by the variance of the terrain assisted innovations. This is because the Kalman update is a weighted average between a state prediction and new sensor measurements, and as such this process will not be instantaneous. Instead, each update will counteract a certain portion of the accrued error. Considering the possibility of a bias in these innovations, it is likely that the variation between successive measurements will be much less than the confidence in the accuracy of each individual innovation. Therefore any measurements in near future frames are likely to be similar to that of recent past frames. It would hence be beneficial to search for spline match data associations around this point. The size of the search space must be chosen to cover any likely position errors of the vehicle or map. This area may therefore be chosen using multiples of the position standard-deviation, e.g. a three-sigma certainty bound.

To implement the search space offset estimate, the vehicle is temporarily shifted by the current innovation estimate, before curvature based association is performed. Once a match is found, the resulting position difference is considered to be a deviation from the estimated innovation. Therefore, the true update innovation is the sum of the search space offset prediction, and the residual map shift.

The search space offset estimate can itself be updated, by considering the innovation produced by the
curvature association algorithm. Furthermore, the Kalman filter update step shifts estimated parameters by a weighted average of the present state variance and the sensor variance. As the shift estimate is a predictor of the residual of this match, it follows that the variance of this measurement will be the current uncertainty of the vehicle position. The innovation itself will be the residual map shift, with a Jacobian sensitivity matrix, that simply reflects an identity relationship with the vehicle position.

As the vehicle moves, it is expected that the database bias will slowly change. For this reason, a prediction model must be employed to slowly increase the variance of the search space estimate. This can be performed by comparing the distance the vehicle has moved, to a reasonable baseline distance over which the database bias changes. The variance may be therefore increased, until it reaches a predefined maximum magnitude, which encompasses the likely map biases.

### 6.4 Feature Estimation Area

An intrinsic part of the SLAM process is the initialisation of new features into the state vector, i.e. the mapping step. As a consequence of this, the state vector increases in dimension, resulting in an accumulation of parameters undergoing estimation. This will have a significant impact on the computation time required to perform each Kalman update, and also adds to the complexity of data associations due to an increase in the number of possible match candidates. A solution to this problem is to remove distant features from the state vector, pausing their estimation. This can be justified by considering the sensitivity of feature positions outside the field of view of the camera. Considering that the inertial measurement uncertainty decreases the coupling between vehicle and feature variance, it follows that the effects of vehicle state updates on features which have not been viewed for some time, will be minimal. It therefore follows, that once the vehicle has moved sufficiently far enough away from a certain feature, any innovations affecting new features, or the vehicle, will not affect the older feature estimate. Therefore there is little point in continuing to estimate its position.

The drawback to removing features from the state vector is a decrease in solution accuracy. This is due to the cross correlation terms in the state covariance matrix containing information about the vehicle and other features. It is therefore necessary to find a compromise between solution accuracy and computation time. Such a compromise may be achieved by analysing the distance between features (e.g.
spline nodes) and the centre of the camera, relative to the field of view. This distance can be defined as a camera view factor ($F$), where any value less than one, corresponds to a feature that is currently visible in the camera. Larger values, therefore, denote features which are outside the camera frame, with the magnitude of $F$ suggesting how much larger the camera field of view would have to be, in order for this feature to become visible (see figure 6.9). It then becomes reasonable to place a threshold on $F$, and use this value to remove any feature which is sufficiently distant from the camera view.

Figure 6.9: Definition of the camera view factor ($F$) with state estimates (blue) and known feature database (magenta)

The camera view factor can also be used to decrease the computational expense of the spline based terrain aiding step. For example, the result of using a Kalman filter is that the pose of the vehicle is always known to an estimated uncertainty bound. This makes it possible to predict which database features will likely be visible at any particular time. Features which are sufficiently far enough outside the predicted camera view may be ignored, as they are unlikely to be currently visible. This feature search space may subsequently be governed using the camera view factor $F$, and may be dynamically adjusted based on the present vehicle pose uncertainty. Similarly, the number of SLAM estimate features used for data association can be limited to improve TAN computation time. Considering that the more features used in data association increases the time required to perform the spline matching algorithms, it is imperative to limit the features used to those which are most likely to be matched. This can be done by considering features which are either currently visible or recently viewed, as these features will have the strongest
correlations with the vehicle position.

There are three separate situations where the camera view factor $F$ can be used to improve computational efficiency. Firstly, by dropping features from the state vector once they are no longer sensitive to new innovations ($F_K$). Secondly, the spline TAN database search space can be limited to only nearby features, which are likely to be visible ($F_D$). Finally, the SLAM features used for association with known database features, can be limited to those most recently viewed ($F_V$). These three thresholds are also mostly independent of each other, however as $F_V$ is used to trim the state estimate features, this threshold cannot be larger than $F_K$.

### 6.5 Compensating for Water Level Changes

One of the most clear visual markers available with which to aid navigation is the edge of a water body, such as a lake or river. A significant source of error, in the edge shape and locations of a pre-processed feature map, is caused by inconsistencies in water level. For example, if navigating over a lake or river (as in the case of the data presented in this thesis), should the water level change between missions, the slope of the riverbank will cause the water edge to shift. These position shifts may result in a horizontal bias on the terrain assistance innovations. These shifted innovations will cause the vehicle pose estimate to converge to a biased solution, potentially degrading the pose estimate. This shift can be characterised and accounted for, through consideration of the riverbank slope. This information can be obtained through the use of a digital terrain elevation map (DTEM). As a DTEM is effectively a grid of latitude and longitude positions with a terrain height above a datum for each, the terrain gradient can be easily found by differentiating this information. It is convenient to perform this in an orthogonal coordinate system with equal units in each direction, such as local vertical, local horizontal (LVLH).

Once the local terrain gradient has been determined, a point identified as lying on the water edge may be adjusted based on the relative water level $\Delta h$ between current levels, and when the edge map database was constructed:

\[
\begin{bmatrix}
\Delta P_x \\
\Delta P_y 
\end{bmatrix} = \Delta h \begin{bmatrix}
\frac{\partial h}{\partial x} \\
\frac{\partial h}{\partial y}
\end{bmatrix} / \left\{ \left( \frac{\partial h}{\partial x} \right)^2 + \left( \frac{\partial h}{\partial y} \right)^2 \right\}
\]  

(6.11)
where \([\Delta P_x, \Delta P_y]\) is the estimated shift in location of this water edge point. \([G_x, G_y]\) are the terrain gradients relative to directions \(x\) and \(y\) defined by the coordinate system. In the case of water level heights, this information can be found for some water bodies (such as fresh water dams) using available data such as online sources.

### 6.6 Water Level Estimation

As established in section 6.5, a variation in water height can affect the location of a lake or river edge. This can cause localisation problems if the edge is assumed to be accurate. However, differences in positions of viewed edges and predicted edge features, can be used to estimate the water height, by introducing a water level parameter to the Kalman state vector. The gradient of lake edges or river banks can be estimated by using a DTEM, and can be used to provide cross-correlations between terrain assistance innovations and water level. This water level estimate, when combined with terrain slope information, can be used to adjust the terrain feature map during flight. This process facilitates self-correcting of the feature map, decreasing the position biases resulting from inconsistent water levels. Furthermore, it provides a method for estimating the amount of water in remote lakes and rivers, which may be used to quantitatively assess riverine ecosystem health from the air.

Once a terrain association has been made between visual features and the known feature database, individual closest point innovations can be used to estimate any differences between the water edges known and viewed water edges. Local terrain slopes can then be utilised to determine a water height offset which best aligns these two edges:

\[
I = H_s - H_p
\]

\[
\partial H = \begin{bmatrix} \frac{\partial z}{\partial x} & \frac{\partial z}{\partial y} \end{bmatrix}^T
\]

\[
\Delta h = (\partial H^T \partial H) \backslash \partial H^T I
\]

where \(H_s\) are the detected water edge points, and \(H_p\) are corresponding closest points on the known database splines. Partial derivatives \(\partial z / \partial x\) and \(\partial z / \partial y\) are the local terrain slopes in \(x\) and \(y\) at each point in \(H_p\).
The main problem with this method is with the conditioning of particular terrain shapes. For example, should the visible terrain be a simple angled plane, then any position error of the vehicle will directly affect the estimated water height. Furthermore, as the water height itself affects the location of the known features, and the feature position affects the terrain assistance position updates of the vehicle, the state estimates can become unbounded and drift. For this reason, it is important to ensure that only innovations constructed from sufficiently robust data are fused. This can be done by examining the individual local terrain slope distribution, and quantifying the variation in these points. Should the variation be large enough, the information can be fused to estimate the water height with confidence. Terrain slope variation can be evaluated by comparing the mean terrain slope with the standard deviation of the set. Should the mean be less than the standard deviation, it can be inferred that the estimate is reasonably independent of vehicle position drift.

6.7 Results

Verification of the algorithms presented in this paper is performed using post-processed data, from a flight over Lake Burragorang and Warragamba Dam in NSW, Australia. The aerial imagery was recorded at 5Hz during this flight using an Ethernet colour digital video camera, with a resolution of 1024 by 768 pixels, and horizontal field of view of around 47.5°. All image frames, IMU and magnetometer measurements are time stamped on collection for synchronisation purposes.

A 400 second period of the flight is used for algorithm verification, the results of which are presented. This section was chosen due to extensive forest/water boundary content, along with periods where no boundaries are visible. All calculations and parameter estimation is post-processed on a 4.2 GHz Intel i7-4770K CPU.
6.7.1 Spline Feature Database

A series of 46 images, of resolution 2560 by 1357 pixels, were recorded from Google Earth in a tiled pattern covering a section of Lake Burragorang. These images, using JPEG compression, sum to a total of 19.2 MBytes of data. Once processed via texture segmentation, splines are fitted to detected edges. These splines constitute the feature database. This database consists of spline node coefficients and the border texture types, and only requires 82.9 kBytes of storage space. This represents a compression ratio of 237:1.

The resulting splines from the Google Earth image processing are shown in figure 6.10. These are overlaid on Google Earth imagery for comparison. It can be seen that the resulting splines clearly capture the essential shape of the lake edge without undue detail. Only the lake edges relevant to the vehicle path were processed for the purposes of this demonstration.

Figure 6.10: The spline feature database processed using Google Earth imagery is shown to capture the true terrain shape.
6.7.2 Statistical Analysis

The errors in the spline based TAN innovations can be quantified through forcing the vehicle to the GPS/INS position solution, while performing mapping and data association. The resulting innovation distributions are outlined in figure 6.11. These distributions can be seen to be approximately Gaussian, suggesting the use of a Kalman filter for data fusion is valid. Unfortunately, it can also be seen that there is a clear bifurcation in the data, specifically in the east-west direction. This can be explained by a combination of water level and Google Earth imagery bias errors. Despite these systemic errors, the spline TAN innovations are sufficiently accurate, demonstrating an error standard deviation of around 10 metres.

![Figure 6.11: Analysis of TAN innovation distributions both for north-south and east-west directions.](image)

6.7.3 State Estimation Area and Sensor Variance

To demonstrate the robustness of the spline based terrain aided SLAM algorithms presented in this thesis, the simulation was run for a range of different sensor uncertainties, and camera view factors ($F_K$, outlined in section 6.4). The uncertainty (or sensor variance) controls the reliance on visual SLAM innovations, with increased values resulting in additional reliance on inertial measurements, and smaller values on visual innovations. The state node drop factor determines when node states are removed from the state vector, and no longer estimated after moving sufficiently far outside the camera frame. This analysis was performed to determine the effects on computation time and accuracy of varying these parameters. It also allows a direct comparison between spline SLAM solutions, and the resulting increase in state estimate accuracy, after the addition of spline based terrain assistance.
Figure 6.12 part a compares spline SLAM with and without spline matching terrain assistance, and shows the accuracy of the solution estimate when sensor uncertainty is increased. The solutions are averaged over a range of different camera view factors (between $F_K = 1$ and 5). For the spline TAN case, the areas $F_D$ and $F_V$ are assumed to be equal to $F_K$. The standard deviation of the solutions is also shown, which gives an indication of the solution robustness. It is interesting to note that with sensor uncertainty below 4°, the unassisted SLAM estimate accuracy deteriorates quickly, as does the variation in the solutions. This can be attributed to over-reliance on visual updates resulting in large update magnitudes. This in turn results in a highly chaotic process, as small changes in the state may affect data associations, potentially leading to large changes in the estimate. At sensor uncertainties above 4° the solution slowly degrades, however the state estimates are much more consistent. This is due to the solution relying less on the visual updates, and instead tending towards simple dead-reckoning, as the sensor variance increases. This is in contrast to the terrain assisted SLAM results which demonstrate a reasonably uniform, low variance and consistently low error. This suggests that even with very low sensor variances, the terrain aided solution is significantly more robust than unassisted SLAM.

The relationship between solution error and the camera view factor is shown in figure 6.12 part b. The mean and variances are shown averaged between sensor variances ranging between 4° and 10°, to avoid the chaotic region skewing the results. This figure clearly shows that for $F_K$ values between 2 and 3.5, the terrain aided solution accuracy is noticeably superior to unassisted SLAM. It also clearly demonstrates that SLAM accuracy benefits from continuing to estimate node positions after they pass outside the camera frame. However, it also shows that the benefits of increasing this factor above 2 are minimal, and can be detrimental in the case of terrain aiding, above $F_K \geq 4$. 

Figure 6.12: Part a shows the mean RSS position error versus sensor uncertainty assumption, averaged over node drop factor. Part b shows the mean RSS position error versus node state drop factor, averaged over sensor uncertainties between 4° – 10°.
An analysis of the simulation processing times, versus camera view factor, is presented in figure 6.13 part a. It can be seen that keeping more spline nodes in the Kalman state vector has a small impact on the computation time for unassisted SLAM. This is effectively due to increases in complexity of the Kalman prediction and update steps, however does not significantly affect data association times, preventing any serious increases in computational requirements.

Figure 6.13: Part a shows the simulation processing time versus node state drop factor, averaged over sensor uncertainties between $4^\circ - 10^\circ$. Part b shows the mean error vs sensor uncertainty, with constant $F$.

Computational expenses involved in the spline matching step greatly affect the processing time of terrain assisted SLAM, with the simulation time for $F_K = 5$ around 50% greater than that for $F_K = 1$. This is due to the increased search space for data associations, as well as more frequent token and critical point generation, as well as comparisons. This, coupled with the error comparison in figure 6.12 suggests that $F_K$ values greater than around three, are unnecessarily time consuming. The use of high $F_K$ thresholds may also degrade navigation accuracy due to the increased possibility of incorrect associations.

It is important to note that the $F$ values may be different for particular sections of the assisted SLAM algorithm. For example, the spline matching data association can occur over a smaller region than the state estimation. All that is required is for these states to be temporarily masked before data association is performed. To test this, the simulations were re-run, with constant $F_V = 2$ for the matching process of viewed features, and $F_D = 4$ for known database features. The Kalman filter view factor $F_K$ was varied between 1 and 5, as before. The resulting errors can be seen in figure 6.13 part b. This analysis again shows the benefits of the terrain aiding step, with mean errors consistently around 50 m, and an improvement on the accuracy outlined in figure 6.12 part a.
This analysis results in a determination that a $5^\circ$ uncertainty, with a Kalman state $\mathcal{F}_K = 3.5$ is the most efficient combination. The data association terrain aiding step applies a $\mathcal{F}_V$ factor of 2.5 for viewed features, and 3 for the known database $\mathcal{F}_D$. These parameters were chosen to maximise the accuracy of the baseline SLAM estimate, while maintaining a favourable compromise of terrain aided SLAM accuracy and computation time.

6.7.4 TAN-SLAM Accuracy

Figure 6.14 compares the estimated trajectories of the inertial navigation and vision aided solutions, to the GPS/INS ‘truth’. It is evident that the inertial measurement solution cannot be relied upon to provide even a rough estimate of the vehicle position, even after just a few minutes of flight. On the other hand, visual spline based SLAM prevents quadratic inertial drift, providing a bounded trajectory that closely resembles the true path. With the addition of terrain assistance in the form of a pre-known feature database, the trajectory estimate stays tightly bound to the true path, representing a high degree of positional accuracy.

![Figure 6.14: Comparison of trajectories of Spline based TAN-SLAM, versus spline SLAM and inertial integration.](image)

The usefulness of TAN-SLAM is highly dependent on the number of recognisable features which are viewable during the vehicle flight. Figure 6.15 shows where the visual navigation system obtains the information needed to perform both SLAM updates, as well as Spline association based TAN updates. It is clear that as expected, SLAM can only occur when traversing edge boundaries. Furthermore, TAN updates are only applied when traversing boundaries which are specifically represented in the known edge feature database.
Figure 6.15: GPS/INS vehicle trajectory is shown over Google Earth imagery and the database edges used. Also shown are locations where SLAM or Terrain aided updates were made.

The root sum square position error of each estimate is compared in figure 6.16, where the geometric divergence of the dead reckoned solution is clear. The fusion of visual updates through spline based SLAM, results in a dramatic improvement in accuracy, with the vehicle state estimate remaining constrained throughout the flight. With the further addition of absolute positional information from terrain assistance, it is clear drift can be all but negated. Figure 6.16 also shows the distribution of SLAM and terrain aid based Kalman updates. This demonstrates how the filter performance is limited by the availability of useful data; if no edges are visible, quadratic drift occurs. On the resumption of SLAM updates, cross correlation between position and velocity allows a portion of this drift to be negated, however terrain assistance updates are required to nullify drift and properly bound the vehicle position estimate.

Figure 6.16 can also be used to quantitatively assess the relative performance of each method. It can be seen that over the 400 second flight, the spline based SLAM solution generally stays within 150 metres of the true position. Similarly, the terrain aided solution stays within 100 metres of the GPS/INS estimate, however during periods where absolute position measurements are reliably made, error is generally below 20 metres, demonstrating a significant improvement.

The accuracy of the spline TAN solution is obviously highly dependent on the accuracy, reliability and frequency of the spline match innovations. Figure 6.17 shows the terrain aided localisation accuracy evolution, related to the accuracy of the fused TAN innovations. It is clear that the spline TAN solution closely follows any spline associations, and hence also any poor associations, such as at the 200 second
Figure 6.16: Comparison of root sum square (RSS) position errors versus simulation time

mark (circled in red). Figure 6.17 also shows the Kalman uncertainty of the vehicle position over the flight. This demonstrates that spline match innovations significantly decrease the vehicle position variance. It also indicates that the position estimate generally stays within a 2-sigma (95% confidence) bound. It is clear from the figure that periods where the position estimate is outside this bound are the result of poor or biased associations.

Figure 6.17: Spline TAN-SLAM position errors relative to the innovation errors in the spline matching process. The vehicle uncertainty estimate is also presented with a 95% confidence bound.
6.7.5 Computation Time

The computation time required to implement this terrain aided spline SLAM algorithm, is outlined in figure 6.18, where the calculation time is broken down for each frame.

![Diagram showing TAN-SLAM Processing Times]

Figure 6.18: Simple breakdown of processing time required for each frame during the simulation

**Prediction:** The first fraction is that of the Kalman prediction step (grey). This is running at the update frequency of the IMU, or 100 Hz. Therefore, although each individual update does not require much computation time, the sum of the 20 steps between each 5 Hz camera frame is noticeable. The time taken in this step is directly affected by the number of Kalman states, hence when large numbers of edges are being simultaneously tracked, there is a minor increase in processing time. This can cause calculation times to rise to over 20 ms, however when no or few features are visible, the prediction step is completed in around 15 ms per frame.

The prediction step also includes a small amount of overhead, due to the prediction and Kalman fusion of a static pressure measurement and magnetometer. The time taken for this is not shown as it is negligible, at under 10 ms.

**Edge Processing:** The edge processing time (red) can be seen to be approximately constant, at around 20 ms. This constitutes the analysis of a pre-processed texture segmentation result, and the sampling of points along detected boundaries. The outlined computation time for this step does not include the texture segmentation processing itself, which is a highly intensive process. The texture segmentation step alone takes around 300 ms per frame, and is by far the most computationally expensive part of the entire assisted
SLAM process. It therefore must be assumed that this processing will be performed on dedicated image processing hardware, in parallel to the state estimation.

**SLAM Update:** The EKF based SLAM processing time is clearly coupled to the number of visual measurements captured, and the extensiveness of edge features currently being tracked. During periods of dead reckoning with no visual measurements, the computation time for this step can be seen to be minimal. During periods of navigation over visually identifiable edge boundaries, the fusion of new data and initialisation of new features has a large impact on computation time. Figure 6.18 demonstrates that the SLAM update computation time generally stays below 50ms, though in some instances it can peak to over 100ms.

**Terrain Assist:** The high level data association techniques involved in robustly matching spline curves add a significant amount of computation time. This occurs during periods where both visual features are viewed, and terrain features are known to be in the vicinity. It can be seen from figure 6.18 that the time taken to perform this association and to fuse position information into the state estimate is comparable to that of the SLAM process at around 50-100ms per frame. Some uncommonly complex frames can result in processing time spiking to above 100ms.

The frame calculation times shown in figure 6.18 demonstrate that over the analysed section of flight, a constant refresh rate of 5Hz can be relied upon for unassisted SLAM. With the addition of terrain assisted position updates, 5Hz is usually achievable, but the rate may reduce to 3Hz for short periods, when navigating more complex terrain.

Figure 6.19 part a shows a more comprehensive breakdown of the frame requiring the most computation of the sequence. This clearly demonstrates that the sensor appraisal and estimation of the magnetometer and static pressure measurements are performed very quickly, in around 10ms. The matrix calculations involved in predicting the state, and fusing in these simple measurements, is negligible at around 5ms. This is echoed by definition of edge points through sampling points along the detected edges, which requires around 20ms. The Kalman update itself, and the initialisation of new features, require significantly more computation time, at around 65ms and 30ms respectively.

The terrain aiding step is clearly the most computationally expensive part of this operation, adding
Figure 6.19: Part a shows a breakdown of computation time for processing the most computationally expensive frame in the simulation. Part b shows a breakdown of the mean computation time for processing all frames.

around 125ms to the computation time. Most of this processing is expended in the calculation and comparison of tokens, as well as the derivation of Jacobian information required for the Kalman update. The adaptive spline filtering step also requires significant time, as this necessitates scanning each viewed and known spline for inflection points, and iterative re-fitting. Furthermore, the point regression step is also computationally expensive, due to the iterative nonlinear least squares regressions involved.

Figure 6.19 part a represents a worst-case sample. Figure 6.19 part b represents the more typical case and shows the mean time spent performing each part of the simulation. This clearly shows that although the spline matching data association step can be quite computationally costly, it is not generally so. When averaged over all frames in the flight, it becomes clear that terrain assistance is a significant part of the computational expense, however it will not dominate the available resources except in complex environments.
6.7.6 TAN Without SLAM Comparison

The benefit of combining SLAM with TAN over the direct implementation of the TAN methodology is a marked improvement in robustness, frequency and accuracy of TAN innovations. This can be verified by removing the SLAM functionality of this algorithm, and treating each camera frame separately. Figure 6.20 demonstrates the resulting navigation accuracy loss. It is clear that the solution accuracy is almost always superior when SLAM is used. This is expected behaviour, as features obtained from each individual frame contain a significant amount of noise. In contrast, the SLAM process uses multiple camera frames to refine a more accurate portrayal of the feature shapes, improving the quality of feature estimates, as well as the robustness of TAN innovations. Furthermore, from the 200 second mark onwards, the TAN-only solution can be seen to diverge, as spline based data associations no longer occur. This is due to the limited view of individual camera frames, which are further corrupted by solar reflections. Therefore, each frame does not contain sufficient unique information to localise the vehicle. The combination of multiple frames through the SLAM process, however, allows more extensive features with unique information to be assembled, allowing successful TAN associations to be made.

Figure 6.20: Comparison of Terrain Aided SLAM, versus TAN performed without the SLAM process.
6.7.7 Water Level Adjustment and Estimation

As outlined in section 6.5, changes in water level can affect the locations of water edge boundaries, and therefore result in biases in the position innovations from TAN. For comparison purposes, figure 6.21 compares the navigation accuracies of the TAN system, both with and without map corrections for water level. This figure shows that the vehicle position remains bounded in both cases, however it is clear that the use of the edge correction does improve the solution.

![Figure 6.21: Comparison of navigation accuracy, with and without water level based edge feature map correction.](image)

Figure 6.21 shows the evolution of the water height estimate, where the simulation was re-run, initialising the water level difference to zero (i.e., to that of when the Google Earth imagery was taken) with a large uncertainty. It can be seen that when the water height is allowed to evolve, it converges to a value very close to the truth (verified against WaterNSW [119] records on the date of the flight). This demonstrates that this method is viable for both estimating the water level in remote water bodies, and therefore, decreasing the position bias that may result from changing water levels over time.

6.8 Summary

This chapter outlines the development of a spline based TAN extension to the SLAM process. This system has been shown to be capable of constraining position drift to a high degree of accuracy, while remaining close to real-time execution. Spline-based data associations between viewed splines and a known terrain map, have been shown to be robust, allowing confidence in the accuracy of the position innovations.
Systematic errors in the creation of the feature map can result in significant position biases in the navigation system localisation solution. These can be the result of geo-referencing biases in aerial imagery databases, or from the environment changing over time, such as river water level variation. Methods can be employed to decrease the effects of these biases. These can involve passive means such as collecting a-priori information to correct the map before operation, or actively working to estimate these biases during flight.

The intelligent consideration of recent terrain innovations can be used to improve computation speed of the TAN process. The result of this, as well as limiting data association candidates to the most relevant, nearby features, results in a terrain assisted navigation system which is capable of running in real-time. The demonstrated system has been shown to be capable of operating on camera imagery predominantly at a rate of 5 Hz, however frequencies may have to be reduced to 4 or 3 Hz depending on the complexity of local terrain.

The outlined system has the capacity to be capable of constraining the vehicle position estimate to within 20 metres of its true position, while identifiable edge features are visible. Furthermore, as these terrain innovations are relative to an absolute position, this system does not risk solution divergence over time. This therefore makes this system more robust and reliable than a strict SLAM implementation, either with or without range measurements.
Chapter 7

Terrain Profile Assisted Navigation

This chapter outlines a range of different methods by which a digital terrain elevation map can be more fully utilised to constrain inertial drift of an aerial vehicle. The use of a DTEM is highly beneficial to aerial navigation, as they are freely available for the majority of the Earth’s landmass. Furthermore, minimal, if any, processing is required before they can be used for data association.

Three methods of using the DTEM to constrain drift are outlined in this chapter:

1. Optical flow information is used to infer the range to underlying terrain, based on the Kalman estimates of the ego-motion of the vehicle. This information can be used to model the shape of this underlying terrain, which can be associated with the DTEM. This allows absolute vehicle position information to be fused into the Kalman filter, constraining position drift. This method does not require image segmentation.

2. Optical flow information may also be used to estimate the motion of the vehicle. A prediction of the range to the underlying terrain based on the DTEM, can be used to estimate the velocity of the vehicle. This allows velocity to be constrained without any image segmentation.

3. The DTEM can be used to estimate the location of water edges without the use of any a-priori imagery processing. This can be used for association with the SLAM map, resulting in position innovations.
These systems can therefore be used to complement the spline SLAM system, maximising the use of available data, as well as better constraining inertial drift. These systems are outlined in the following sections:

Section 7.1 provides an overview of using optical flow for navigation assistance.

Section 7.2 shows how optical flow can be determined from pairs of temporally adjacent frames. This section also demonstrates how the DTEM can be used to predict the flow, reducing the complexity of the optical flow algorithm, hence helping to eliminate spurious results.

Section 7.3 outlines how optical flow can be used to estimate local terrain contours. This section also presents how these terrain estimates can be associated to a known DTEM.

Section 7.4 demonstrates the use of optical flow for visual odometry, with DTEM data providing range information. Camera transform models are used to improve the accuracy of the method, reducing linearisation errors.

Section 7.5 profiles how DTEM data can be used to predict the locations of water edge boundaries. This information may then be used for data associations with viewed splines from the SLAM process, and used for localisation.

Section 7.6 presents the resulting accuracy of the application of these optical flow and DTEM aids to aerial navigation. The computational expense of these methods are also quantified.

Section 7.7 provides an overview of the outcomes of the work presented in this chapter.

7.1 Optical Flow Overview

As outlined in section 3.8, optical flow is a measure of how quickly objects captured by a video stream move in the camera frame. This motion can be the result of any one, or a combination of three rates: Motion of the target object ($\vec{v}_f$), motion of the camera ($\vec{v}_c$), or rotation of the camera ($\vec{r}_c$). Of these, the optical flow magnitude is dependent on the distance between camera and feature for the first two cases, but not for camera rotation rates. A linear approximation to this flow is expressed in section 3.91.
It is clear from equation 3.91 that in order to determine one of these rates or the separation distance, a large number of parameters must be known. Fortunately, the outlined navigation system presented so far in this thesis, either directly measures ($\vec{r}_c$) or estimates ($\vec{v}_f$, $\vec{v}_c$, $R$) all of these parameters. The camera rotation rate is directly detected through the output of a 6-axis IMU, and therefore can be accurately characterised. The motion of the vehicle is estimated by the Kalman filter, whereas the object features viewed are assumed to be stationary. This allows the range, or distance between feature and object, to be used in estimation of any vehicle motion. The range itself can be approximated through the DTEM ray-casting technique outlined in section 5.2.

The optical flow equation (eq. 3.91) therefore permits these direct and inferred measurements, to be used to constrain drift in a number of different ways. These methods involve using the equation to estimate one of the parameters, which can then be compared to the existing estimate, and used as an innovation. For example, range and the optical flow measurement can be used to infer vehicle velocity, which may then be used as an innovation to improve the Kalman velocity estimate. This example therefore would result in visual odometry.

Similarly, the vehicle velocity estimate can be used to determine range errors. The height above a reference point (for example mean sea level) can be assumed to be $a$, and the local terrain height beneath the vehicle $h$ can be estimated using the DTEM. For this point on the terrain directly below the vehicle, the distance ($R$) between vehicle and terrain point can be estimated to be:

$$ R = a - h $$ \hspace{1cm} (7.1)

If it is assumed that there is an unknown error ($\Delta h$) in the terrain height, the true range $R_t$ can be rewritten as:

$$ R_t = a - h + \Delta h $$ \hspace{1cm} (7.2)
Equation 3.91 can be simplified through the assumption that visible features are stationary, and rewritten:

\[ F = \vec{r}_c + \frac{\vec{v}_f}{R_t} \]

\[ F = \vec{r}_c + \frac{\vec{v}_f}{R + \Delta h} \]  \hspace{1cm} (7.3)

\[ \Delta h = \frac{\vec{v}_f}{F - \vec{r}_c} - R \]  \hspace{1cm} (7.4)

therefore resulting in a measure of the errors between range based on ray-casting, and via optical flow.

It is important to note that these equations are approximations, based on the assumption that range is a constant. As this will clearly not be the case, these equations act as a simple summary of the generalised methods outlined in the following sections.

### 7.2 Optical Flow Calculation

Video imagery by its nature, combines both a spatial component in pixel locations and values, with a temporal component indicating how these values shift and change between frames. Optical flow is a measure of the movement of pixel values with time, i.e. the direction and speed objects in the video images are moving. This is performed by analysing consecutive video frames to determine an image warp (image co-ordinate re-interpolation), which acts as a transformation between the two frames. This re-interpolation map may then be considered to be the optical flow, describing how far particular regions of the image have moved between frames.

A significant drawback of the use of optical flow information is the computational expense required in its determination. Efforts to streamline and simplify this process are therefore highly beneficial to any real-time processing goals. In the case of aerial imagery, it can generally be assumed that captured motion will occur in a single, predictable direction in any particular camera frame, depending on vehicle motion and rotation. Therefore, the process of determining optical flow from camera imagery can be simplified, as the result can be pre-empted. This means that instead of considering every possible optical flow result, only deviations about expected values need to be considered. This constriction of the flow search space not only decreases the computational expense of the operation, but also allows outlier measurements to be more easily identified and subsequently removed.
The computational expense of optical flow calculation will also place limits on practical frame rates, unless dedicated processing hardware is considered. The drawback of this approach is that the greater movement generated between successive frames will amplify the nonlinearity of the process, adversely affecting the result. The result of this is that the equations outlined in section 7.1 become less accurate, with decreasing frame rates. Therefore, to sidestep this issue, the full non-linear equations of vehicle motion and camera transformations may be included into the relationship between optical flow and vehicle motion. Further benefits of this process generalisation is that the optical flow methodologies can be made to more closely reflect the mechanics of SLAM, by tracking arbitrarily placed points. An outline of the application and background for this nonlinear flow generalisation is covered in this section.

### 7.2.1 Optical Flow Prediction

Due to the use of a high precision IMU and SLAM filter to track the vehicle state, it can be assumed that the vehicle pose and motion are known to a reasonably high accuracy. Furthermore, as the DTEM data is known, a prediction of the apparent optical flow in a particular video image, can be deduced. Based on the current pose estimate, a grid sample of points in the camera frame are projected to the DTEM, using an inverse camera transform model (figure 7.1). The vehicle motion estimate can then be used to reconstruct the previous pose of the vehicle, at the time the penultimate video frame was captured, by employing a Kalman prediction with a negative time step. Finally, the position in the camera of these sampled points on the DTEM, at this new past vehicle pose, is determined using the algebraic camera model. This process gives the in-camera position differences, between the original point sampling at the vehicle current pose, and at the previous camera frame pose. Furthermore, the use of both the algebraic camera and state process models, capture any nonlinear effects.

### 7.2.2 Flow Update

Once a prediction of the optical flow is calculated, this can be used to warp the previous image, into a prediction of the new image. A simple optical flow algorithm is then performed to compare the true new image and the image prediction, to determine any residual differences. As any differences can be assumed to be a result of errors in the DTEM, the state estimate or of noise, it can be surmised that they
Figure 7.1: Sampling points onto a DTEM can be used to estimate optical flow measurements. will be small. This suggests that large differences are artefacts from errors in the optical flow calculation algorithm, and can be ignored. For this reason, a simple template matching method is used to calculate the optical flow, consisting of identifying the differences between the two images at a range of pixel offsets, both horizontally and vertically. The range of these pixel offsets is chosen to cover the maximum likely flow result, in order to save computation time.

Figure 7.2: Template matching can be used to determine optical flow between images.

At each pixel offset, the two images are directly compared, with the absolute difference between each pixel recorded in a variance map (figure 7.2). These maps are then filtered with a small Gaussian kernel to remove noise. A rough nearest pixel solution may then be calculated, by individually selecting the offsets which give the smallest variance for each pixel. This estimate is then refined for each pixel, by fitting parabolic functions to the variances of this offset, and all adjacent offsets (figure 7.3). This parabolic fit can then be used to update the optical flow solution to a sub-pixel resolution, by identifying
the minimum position of each parabolic fit.

Figure 7.3: Using quadratic modelling can improve optical flow calculation accuracy.

7.3 Optical Contour Association

Estimation of a terrain profile can, in theory, allow drift-free navigation through data associations with known terrain contour information. This is clearly dependent on the existence of distinguishable terrain profile features, such as hills and valleys. In the presence of such features, matching techniques between surface objects can be used to determine any errors in the position of a vehicle, which may be used to correct the vehicle pose estimate. The use of terrain elevation contour matching to localise aerial vehicles has been proven to work for decades, since TERCOM [7, 8] in the 1950’s. This section outlines how this idea can be modernised, by replacing active radar range measurements, with inferred range from passive optical flow techniques. This has the dual benefit of reducing the power draw of the system, and facilitating stealthy navigation.

This method can be separated into four sub-sections:

1. Optical flow measurements are used to estimate the range to viewed terrain. This can be performed with consideration of the Kalman velocity estimate of the platform.

2. The range to terrain can be used over successive frames to build a local profile estimate. This helps to maximise the data available for matching, as more terrain than just what is presently visible can be used in the matching process. Averaging out a profile over a number of frames, also helps to decrease the effects of optical flow measurement noise.
3. The profile estimate is matched to a known DTEM to determine an innovation, which represents the error in the current vehicle position estimate. As vehicle velocity errors will affect the range to terrain and hence the height of the profile estimate, this matching process predominantly relies on shape comparisons.

4. Position innovations resulting from this matching process can be fused into the Kalman filter. This will help to correct the vehicle position estimate, the positions of any mapped SLAM features, as well as the position of the terrain profile estimate.

7.3.1 Range from Flow

Optical flow can be used to estimate the range \( R \) to underlying terrain by using the simple equation:

\[
R = \frac{V}{\omega}
\]

(7.6)

where \( V \) is the velocity of the vehicle, and \( \omega \) is the angular rate obtained from optical flow. This method however, may not fully account for the nonlinear motion of an aerial vehicle, resulting in decreased accuracy. Therefore, it is beneficial to fully utilise both camera transform and vehicle motion prediction models, to best estimate range from flow.

This updated method uses the vehicle state \( X \) when the previous camera frame was taken (time = \( t - 1 \)), as well as the present state. The present state can be used to define a vector which points towards an arbitrary point on the ground currently in the camera frame. The optical flow reading at that point, can be used to define a second vector emanating from the previous vehicle state, also oriented towards this point. These two vectors can then be compared, and the point of intersection (or the closest points of approach) calculated, to determine range.

This method potentially raises problems when the update step of the Kalman filter is considered, as it can be assumed that Kalman updates will occur between successive camera frames. These updates may result in position changes, which may significantly affect the baseline distance between states, as outlined in figure 7.4. This diagram shows how using \( X^+_i \) and \( X^+_{i-1} \) may cause errors in the range prediction due to position updates. One solution is to use the a-priori state \( X^-_i \), however this ignores any recently
required information, and therefore still loses some accuracy. Furthermore, if multiple updates are to occur during this interval, a second state estimate must be predicted in parallel to the main SLAM state vector, increasing complexity. It therefore becomes much simpler, more accurate and computationally efficient, to instead re-construct the previous state estimate using a reverse prediction step. This allows both states used to contain all collected information up to the present moment, while simultaneously ensuring that any updates occurring between camera frames do not render the range prediction inaccurate.

### 7.3.2 Terrain Contour Estimate

The optical flow range estimations calculated using the method outlined in the previous section, may be used to infer the shape of viewed ground. However, owing to the nature of the methods used to obtain this information, a large amount of uncertainty exists related to these measurements, due to errors in velocity and the calculated optical flow. For this reason, it is beneficial to iteratively sample terrain contour estimates, and fuse them into a combined estimate. This also carries the benefit of allowing a more substantive area of terrain to be gauged than just what is presently viewed in the camera. This estimation methodology will require a prediction and update step, similar to a Kalman filter. Because of the number of states (each estimated ground height point), an independent linear regression method is preferable to a full Kalman filter, due to computational requirements.
Terrain Prediction

There are multiple methods of predicting the shape of the terrain that is likely to be visible. These are analysed below:

**Earth Fixed Grid:** Perhaps the most obvious method is to create a second DTEM (figure 7.5), using the known terrain as a baseline with a particular uncertainty. Any new terrain measurements may then be used to update this terrain map, using the present state estimate. Associated advantages of this method are the automatic generation of an Earth-fixed terrain map. Furthermore, revisiting areas will allow terrain estimation to start from the last best estimate of the terrain shape. This method however, has the significant disadvantage of culminating in discontinuities in the map after any position updates, or information loss through re-interpolation. This can cause association issues with the known map.

![Earth Fixed Grid Prediction Step](image)

![Earth Fixed Grid Update Step](image)

Figure 7.5: An Earth fixed terrain estimation grid requires no adjustment for Kalman predictions, however it requires re-interpolation after any Kalman position updates.

**Vehicle Centred Grid:** In order to avoid any discontinuity problems, a floating, vehicle centred map (figure 7.6) can be defined. This would then move with the vehicle and encompasses an area covering any likely terrain the camera is likely to view, with any reasonable resolution. The disadvantage of this method is that as the grid moves with the vehicle, the terrain that the grid defines must also be shifted within the grid. This can be performed by an advection step, where the features are re-interpolated onto the new grid, in its updated position. This re-interpolation can be thought of as a filtering step, and as such the terrain shape will degrade over time. Furthermore, the advection step adds extra computation, and in some cases can restrict the terrain grid spacing and vehicle velocities. As the grid is fixed to the vehicle, this method does have the advantage of automatically adjusting the estimated profile location.
should any position innovations occur.

Figure 7.6: A vehicle centred terrain estimation grid requires re-interpolation after every prediction step, however position adjustments for Kalman position updates occur automatically.

**Semi-Centred Grid:** Most of the disadvantages of the vehicle centred grid can be removed through minor modifications to how it is defined. Instead of fixing the grid to be exactly below the vehicle, it remains stationary during Kalman prediction. The vehicle instead is allowed to move within half a grid spacing of the grid centre. Once the vehicle moves outside this bound, the far row or column of the grid is deleted, and a new, opposite row or column is initialised. This returns the vehicle to the centre of the grid. The position difference between the grid centre and vehicle is updated using a simplified process model, governed by the vehicle velocity. The outcome of this is that any vehicle state position updates resulting from sensor fusion will also shift the map grid, preventing any discontinuities. Furthermore, as any re-interpolation or advection is therefore unnecessary, there is no degradation of the map over time. Finally, there are no significant computational penalties arising from this process, as all re-interpolations have been eliminated. Effectively, this combines the advantages of the previous two methods, while removing the subsequent problematic disadvantages. Because of these benefits, and the relative lack of drawbacks, this is the preferred estimation grid methodology.

**Terrain Update**

Once an estimate of the underlying terrain has been made by using the existing DTEM, new range measurements obtained from optical flow can be utilised to update this estimate. This can be performed by modelling the terrain as a 2-dimensional non-uniform rational b-spline, or 2d NURBS. This model allows Jacobian matrices to be calculated describing how points on the surface are affected by the node
coefficient values. This allows the terrain prediction to be updated with new shape information from the optical flow estimates.

This update can be performed using a simple linear regression, weighted by the uncertainties of the terrain estimate, the individual range estimates, as well as the Jacobian sensitivity weightings:

\[
G_{n+1} = \frac{G_n \sigma_n + \sum G_i \sigma_i}{\sigma_n + \sum \sigma_i}
\]  

(7.7)

where \(G_{n+1}\) is the updated NURBS, with the a-priori NURBS denoted by \(G_n\). The a-priori node uncertainties are contained within \(\sigma_n\), with the individual measurement node positions \(G_i\), each with uncertainties described by \(\sigma_i\).

This method loses some accuracy compared to a full 3D Kalman filter step, due to loss of cross-correlations, however it is significantly more computationally efficient. For example, if estimating a 50 by 50 terrain grid, this would require the addition of 7500 states to the Kalman filter, or 3 for each node. The matrix inversion involved in the Kalman update step, will clearly be a highly computationally expensive operation, which will make real-time operation highly problematic. Conversely, simplifying the problem to a one-dimensional weighted mean is comparatively quick, making real-time applications possible.

Figure 7.7: Range inference from optical flow can be used to estimate and update the shape of underlying terrain as the vehicle passes.

The terrain estimate itself must be initialised before the estimation process can begin. Initialisation at
sea-level would be an obvious choice, however this could result in biases over extended periods, as it would be expected that viewed terrain would predominantly be distributed above this level. The result of this would be innovations of non-zero mean. Instead, the local DTEM itself can be used to initialise the terrain estimate. This results in a much higher likelihood that the terrain height innovations will be of zero mean. As the vehicle traverses the terrain, the profile estimate will evolve from the DTEM values, to those suggested by the optical flow information. This is demonstrated in figure 7.7.

### 7.3.3 Contour Association

Once an estimate of the underlying terrain has been made, it can be compared to the known surrounding terrain from the DTEM. As the terrain estimate grid spacing has been chosen to be equal to that of the known DTEM, the height nodes may be directly compared. This is performed in a four step process:

1. **Search Area:** A likely search area is determined based on the present vehicle position state uncertainty, as well as uncertainties in the DTEM [120].

2. **Coarse Match:** The terrain is compared at a sampling of grid spacing offsets contained within this region. This results in a rough offset which gives the best match between the two terrain profiles.

3. **Fine Match:** The terrain is compared at a finer resolution at the best coarse match point, to determine how the match quality or cost is affected by position changes.

4. **Match Variance:** A paraboloid is fitted to the distribution of local match costs, which is used to determine the minimum value and location. A variance ellipse may also be defined based on this minimum cost, which describes the uncertainty of the association. The dimensions of this ellipse may be derived from the local match cost distribution.

   The process outlined above returns a position innovation, based on the location of the minimum difference offset. The variance ellipse of this match gives an indication of the measurement certainty. Finally, the minimum difference value can be used as a match quality rating, providing a quantitative method of judging whether the terrain match is valid.
7.3.4 Vehicle Position Update

The matching metric used to derive the terrain profile associations can greatly affect the performance of the TANS. For example, the use of a weighted mean difference metric, would result in a matching algorithm highly dependent on the vehicle velocity estimate. On the other hand, a weighted standard deviation metric relies more heavily on terrain shape information, but will be more sensitive to noise. This may cause issues when associations are required between terrain profiles which contain minimal curvature information.

In order to retain a simple matching algorithm, while minimising the possibility of poor performance, both a weighted mean, as well as standard deviation metrics are implemented. Weightings are based on the current certainty of the terrain estimate nodes, which helps to prioritise converged areas. Regions with uncertainty above a pre-defined threshold are ignored. These two separate terrain association methods will both return an innovation and a variance, which may then be combined. Furthermore, the present velocity state certainty can be used to dynamically adjust the weighting between these two innovations. This is necessary, as the association based on the mean terrain difference is highly dependent on vehicle velocity. Therefore, a higher certainty in the state velocity estimate will allow greater confidence in the mean difference method result. Once a final innovation and variance have been resolved, this information can be fused into the Kalman state, resulting in a vehicle position update (figure 7.8).

Figure 7.8: The terrain estimate may be matched to the DTEM, resulting in a position correction innovation.
7.4 Visual Odometry

As well as position updates through visual terrain contour association, the combination of optical flow and a DTEM can be used to estimate the velocity $V$ of the vehicle. This can be seen by reviewing, then rewriting equation 7.6:

$$V = FR$$  \hspace{1cm} (7.8)

where $F$ is the optical flow, and $R$ the predicted range to the viewed profile. However, due to the nonlinear motion of the vehicle, this process may be generalised through the use of nonlinear prediction and camera models.

7.4.1 Velocity Update

The apparent optical flow of terrain in the camera, is governed by the motion of the vehicle, as well as the range to the viewed terrain. This is demonstrated by the relationship outlined in 7.6. This allows the vehicle velocity to be inferred from the vehicle rotation, range to terrain and optical flow. This can be performed in a three step process:

1) **Present Frame Sampling**: An array of points in the present camera frame is sampled and projected onto the DTEM, using the current vehicle state estimate.

2) **Past Frame Sampling**: A second array of points is adjusted in the camera frame by the calculated optical flow. These are projected to the DTEM, from a state estimate obtained using the reverse prediction method outlined in section 7.3.1.

3) **Sample Comparison**: The points in these two sample sets are compared (figure 7.9). Any mean difference can be used to update the vehicle velocity, where as the distribution of position offsets can be employed as an indication of measurement uncertainty.

This process therefore results in a velocity innovation and variance, which may be fused into the
Kalman update step.

Figure 7.9: The movement of sampled points, based on the optical flow measurements, can be used to estimate errors in the vehicle velocity estimate.

The drawback of this method is a clear dependance on the estimated range to terrain. The range is also highly dependent on the position estimate of the vehicle, when flying over mountainous terrain. This dependence leads to the possibility of compounding position errors with poor velocity updates. To minimise this possibility, optical flow velocity estimates are only performed after a visual terrain contour match has been made. This allows the vehicle states used for the sample point projection to be temporarily adjusted based on this information, limiting the dependence of range estimates on position drift.

### 7.5 DTEM Water Edge

A DTEM can be used to estimate the boundaries of any potential bodies of water, by analysis of areas which demonstrate negligible height variation. This can act as a crude terrain aid, where the predicted edge features from the SLAM process, can be matched to these boundary estimates. Any associations which result from this process can provide a simple, computationally inexpensive method of localising the vehicle, using only widely available DTEM sources.
7.5.1 Water Edge Estimation

The SRTM Water Body Data (SWBD) [121] is a combined, tiled dataset of water body edges derived from the SRTM maps. Alternatively, the SRTM data can be used to create a similar, bespoke dataset by interpolating water edge points at a designated water height. The benefits of this method over the direct use of the pre-processed SWBD, is that some level of control is retained over the mapped water height. This is especially relevant as the water edge locations of many water bodies can change significantly, due to changes in water level. Therefore, edge locations may be revised should the water level be known, or updated dynamically if the water levels are to be estimated (such as in chapter 6.6).

Figure 7.10 describes this process, where interpolation between DTEM points can be utilised to define the water boundary edge points, at any chosen water height. Differences in position between any SLAM estimated edges, compared to these DTEM edge points, can then be used as an innovation to help localise the vehicle.

Figure 7.10: Flat areas of the DTEM can be identified as water. Knowledge of the water height can be used to adjust the edge points, which can be associated with viewed edges to provide position innovations.

7.5.2 Terrain Water Edge Innovation

This dataset, which results from DTEM processing, consists of a series of edge points along any local water boundaries. These edge points can be associated with the current SLAM feature estimates, by determining the closest points on these splines, to each DTEM edge point. An innovation that best overlays the splines and the DTEM water edge points, may then be derived through an iterated convergence method. Figure 7.11 demonstrates the resulting innovation, which when applied improves the overlap between the
two data sets.

![Figure 7.11: The viewed edge splines may be matched to the DTEM water edge point estimates to calculate position innovations.](image)

Once the association has been calculated, a representative uncertainty of the match may be taken by considering the relative gradients of the edge features. This is necessary as conceptually matching two straight sections, will result in a large amount of uncertainty parallel to the edges (Figure 7.12). Conversely, associating edges which have sufficient curvature, will result in a high confidence measurement with a small maximum uncertainty. This can be approximated by designating uncertainty ellipses, which exhibit a small variance perpendicular to the local edge gradient, and high variance parallel to the local gradient. These ellipses may then be combined into a final match variance ellipse, which can be weighted based on the variance of the terrain map.

![Figure 7.12: The amount of curvature in matched regions will affect match variance.](image)

This ellipse may also be updated by the distribution of individual point-to-point match innovations, demonstrated in figure 7.13. As a high quality, accurate match will result in small residual point innovations, the standard deviation of these associations will be small. Conversely, a less definitive shape
match will result in larger residual point innovations, and as such these matches will exhibit a larger standard deviation. The combined variance of the match must therefore be a function of the DTEM uncertainty, the quality of the association, as well as the general shape of the matched curve.

![Figure 7.13: Variance in point match innovations will affect match variance.](image)

The derivation of the match uncertainty allows this innovation to be fused into the Kalman state filter as an absolute position measurement. This will help to converge the vehicle position estimate, as well as the placements of all mapped features.

### 7.6 Results

Verification of the visual navigation algorithms outlined in this chapter, was performed using the Lake Burragorang flight, which was post-processed using a 4.2 GHz Intel i7-4770K CPU. This section outlines the resulting accuracies of the visual navigation methodologies outlined in this chapter, as well as analysing the computation time required by the system.

#### 7.6.1 Verification of Optical Flow Aids

The optical flow based navigation aids described in this chapter, can be verified through the use of the GPS/INS solution. By setting the vehicle state to these ‘true’ values, it can be assumed that the magnitude of any resulting innovations from either C-TAN, or odometry, should equal zero. This can therefore be used to quantify the accuracies of the presented systems.
Terrain Profile Estimation and Mapping

During the post-processing GPS/INS simulation, terrain height is estimated, and used to assemble a map of the terrain below the vehicle trajectory. This map can then be compared to both a geo-referenced aerial imagery database, and SRTM information. Figure 7.14 shows a false colour representation of the mapped terrain estimates, overlaid on aerial imagery. This shows that the resulting terrain map does manage to successfully represent the true terrain. It can be seen that false colour blue (representing heights above sea level of approximately 100 metres) are predominantly located near the water edge. Areas of false colour red (representing heights of up to 500 metres) are seen to represent the tops of cliffs. Variations in the colours follow the shape of the terrain, such as valleys and hills.

Figure 7.14: Mapped terrain profile, overlaid on aerial imagery. False colour shows estimated ground height (blue = 100 m, red = 500 m).

The errors between the estimated terrain map and the SRTM are quantified in figure 7.15. This shows that the terrain height error approximately follows a normal distribution about a zero mean, and the majority of estimates are within 50 metres of the DTEM. The mean of this distribution is 5.6 metres, which is most likely due to the existence of trees in the imagery, as these are not represented by the DTEM. The standard deviation of these errors is 23.9 metres. It is important to consider that the DTEM itself is not completely accurate, with errors of 10 metres likely [120].

Terrain contour associations between the estimated terrain profile and the DTEM during this process are shown in figure 7.16. It is clear that the contour based data association method produces a significant number of position innovations. These matches can be significantly biased, often exhibiting over 100
metres of error. Despite this, these position innovations will result in the bounding of position drift, although likely to a lower accuracy than spline TAN-SLAM (chapter 6).

Visual Odometry

The use of the GPS/INS navigation solution can also be used to verify the visual odometry method. The results of this are shown in figure 7.17, which shows that the innovations appear to be distributed around zero. The innovations are also mostly within $5 \text{ m/s}$ of the GPS/INS truth.

7.6.2 Navigation Implementation

In order to verify the functionality of the C-TAN system, this lake imagery sequence was processed both with, and without, consideration of visual odometry. The resulting vehicle trajectories are shown in figure 7.18. This clearly illustrates that the odometry solution more closely follows the true trajectory of the
Figure 7.17: Errors in visual odometry measurements, both forwards ($u$) and sideways ($v$).

vehicle. Without the visual odometry aid, the navigation solution tends to diverge in a lateral direction.

Figure 7.18: A comparison of the vehicle trajectories of C-TAN both with, and without, visual odometry.

The resulting localisation accuracies can be seen in figure 7.19 for C-TAN both with, and without, odometry. This figure shows that the addition of visual odometry, does not appear to significantly improve the position error that can be expected from C-TAN. It does however improve robustness, as the vehicle position solution is more stable.

The benefits of visual odometry can be seen more clearly in figure 7.20, by comparing the errors in the predicted vehicle velocity components. This figure shows that visual odometry noticeably improves the precision of the Kalman velocity estimates, for C-TAN. Part $a$ of this figure demonstrates that some biases can still arise, however for the most part, the velocity estimate is more accurate. This is especially clear for the sideways component (part $b$), where the lateral velocity in the odometry fusion case can be
seen to be tightly constrained.

Visual odometry improves the system performance of C-TAN, as the velocity information garnered from direct position estimates is comparatively weak. Successive position innovations can be used to estimate velocity, but this method will result in a lot of noise (as velocity is a derivative of position, which amplifies noise) and therefore, will result in large variances. More importantly, as the terrain profile estimate is dependent on vehicle velocity, position matches can easily be biased by velocity errors. This also results in difficulties estimating velocity.

The C-TAN system must therefore be considered to be a combined process with visual odometry. This odometry is used to estimate the vehicle velocity and the terrain profile, which is then used for position innovations. These associations are, in turn, used to limit drift both in position, and in velocity.

**Computation Time Analysis**

The benefit of navigation using optical flow techniques, over spline based SLAM, is that no texture segmentation is required. This computationally expensive step is replaced with the calculation of optical flow. The optical flow algorithm is also moderately computationally expensive, however only requires around 100 ms to complete (as shown in figure 7.21), compared to around 300 ms for image segmentation. This improvement in computation time however, is let down by the significant processing requirements of the surface modelling and estimation, required to resolve the local terrain profile shape. The actual
a) Longitudinal (Forwards) Velocity Error

Figure 7.20: Comparison of velocity components both longitudinal (a) and lateral (b) for C-TAN with, and without, odometry.

A more comprehensive breakdown of processing times is outlined in figure 7.22. This shows that the calculation of optical flow requires just over 100 ms. This includes utilising the range to terrain to estimate the visible flow, before template matching is used to update this flow estimate. Other elements of this section require less time, such as the prediction of range to the DTEM, and fixing the optical flow solution to remove outliers and otherwise poorly conditioned regions. Other algorithms which require significant time are the determination of the errors which would arise from using a linear approximation, and finally the use of vector methods to estimate range to viewed terrain. These algorithms are predominantly stable with respect to the use of different frames, with minimal variations. In contrast, the computation for fusion of this optical flow and range data, is significantly more varied. The terrain estimate update dominates this section, requiring a large amount of computation time (up to 140 ms) to refine the profile estimate.
Figure 7.21: Trace of the processing times required per frame for optical flow assisted navigation.

The contour matching and visual odometry steps themselves require significantly less time, around 40-50 ms each.

Figure 7.22: Comparison of average frame time breakdowns (part a), as well as the times spent analysing the most computationally expensive frame (part b).

7.6.3 Comparison with Spline SLAM

As the contour-based TAN (C-TAN) system is comparatively fast, it is beneficial to compare this to the spline SLAM navigation system, presented in chapter 5. Both of these systems can operate in near-real-time. Figure 7.23 outlines the navigational accuracy of these two systems. It can be seen that
the two methodologies are for the most part comparable, although spline SLAM is more precise. This comes at the cost of computational expense, as spline SLAM requires an average of around 320 ms per frame, including image segmentation, whereas C-TAN requires around 200 ms on average. A similar difference can also be observed for the most computationally expensive frames. Therefore this increase in navigational accuracy from spline SLAM requires around 50% more computation time, over C-TAN.

Figure 7.23: Comparison of C-TAN plus visual odometry, and spline SLAM. The time locations of optical flow aid innovations are also shown.

Figure 7.23 therefore suggests that spline SLAM is more accurate than C-TAN, despite this optical flow based system returning absolute position innovations, which spline SLAM does not do. Reasons why this is the case include the use of the low resolution SRTM database, which only contains data sampled at spacings of around 30 metres. This therefore, does not capture the terrain shape as well as it otherwise could.

Further refinements to the C-TAN system could be gained from improvements in computational efficiency. Running this system on dedicated hardware could result in significant speed improvements, allowing higher frame rates to be used, and a higher sample point density. This would act to enhance the accuracy of the system.

7.6.4 Terrain Based Water Edge TAN

Fusion of the C-TAN system with spline based SLAM, can significantly improve the accuracy of the system, however does so at a greatly increased computational cost. As previously outlined, optical flow
aids require up to 330 ms per frame, and spline SLAM can demand over 450 ms per frame. Therefore it can be expected that the combination of these two systems can necessitate around 800 ms per frame when operating over complex environments. Fortunately, the extension of this system to include DTEM based water edge associations is straightforward, requiring a negligible amount of additional computation. This therefore results in two separate systems for constraining velocity drift, as well as two separate systems for restricting position drift. Considering this computational expense, and the amount of data fusion, it is logical to compare this system with the spline TAN-SLAM system outlined in chapter 6.

Figure 7.24 compares the navigational accuracy of spline SLAM fused with all of the DTEM aids, with fusion of an edge curve database (spline TAN-SLAM). It is clear that the use of spline TAN-SLAM is preferable to optical flow methods, resulting in a significantly lower position error. Despite this, it is evident that the addition of DTEM edge estimate associations significantly improves the performance of C-TAN. This system produces localisation solutions which are predominantly within 100 metres of the GPS/INS truth. Spline TAN also exhibits maximum errors in this example of around 100 metres, however demonstrates superior localisation over C-TAN, whenever curve matches are made.

![Position Error Evolution of DTEM Aid TAN−SLAM vs Spline TAN−SLAM](image)

Figure 7.24: Comparison of spline SLAM fused with all DTEM aids, versus fusion with known spline database TAN innovations.

### 7.7 Summary

This chapter outlines the development of optical flow based visual navigation aids, which utilise freely available DTEM information. This system is modelled on the TERCOM navigation aid, however employs
a monocular camera in place of an active radar range-finder. DTEM data is also used to estimate the location of water body boundaries, which can be used for absolute position innovations based on associations with spline SLAM features. Such a system can therefore fully localise a vehicle using only a raw DTEM, without the need for a-priori processing of an edge feature map database.

An exclusive implementation of the C-TAN system has been shown to demonstrate navigational accuracies approaching those of the spline based SLAM system outlined in chapter 5. Although this system is less accurate than spline SLAM, it exhibits a number of benefits. Firstly, spline SLAM relies on image segmentation, which requires a-priori knowledge of the environment in the form of texture training data. Secondly, image segmentation is very computationally demanding, whereas the optical flow based C-TAN system is noticeably less intensive. These benefits come at the expense of navigational accuracy, as relative spline SLAM produces a superior localisation solution. This is despite the absolute position innovations provided by the C-TAN system, which would conceptually be preferable to a reliance on relative measurements. This discrepancy stems from the relative ease by which vehicle position or velocity state errors can bias measurements inferred using optical flow information.

The computational analysis presented demonstrates that this system in its current state, is not capable of running in 5 Hz real-time. However the system should comfortably run at an update frequency of around 3 Hz. Instead, many further optimisations could be made to these algorithms to improve computation speed, which would result in real-time operation. For example, specialised hardware already exists to perform optical flow in real-time for UAV systems [122], as well as for optical mouse sensors. This could be used to save a significant amount of computation overhead. Furthermore, as all of the involved algorithms are highly parallel, GPGPU (general-purpose computing on GPU) hardware could be used to boost computation speed significantly.

The fusion of spline SLAM with C-TAN results in a much more accurate solution, however comes with a heavy computational expense. This system is broadly comparable in accuracy to the spline TAN-SLAM method presented in chapter 6, although it does demonstrate noticeably lower accuracy. The benefits of this system are that no significant pre-processing of data is necessary, as only a DTEM is needed. This DTEM is easy to obtain, and any processing required can be performed on-board, during flight.
Chapter 8

Combining Fusion Methods and Real-Time Analysis

This chapter provides a comparative summary of the relative accuracies of the visual navigation aids outlined in chapters 5, 6 and 7. It also endeavours to show the resulting accuracy of a combined fusion of all these navigation aids into a single solution.

In addition, this chapter will present a simple method for ensuring real-time operation, by dropping frames to adaptively adjust the Kalman update frequency. A spline TAN-SLAM example is shown, demonstrating that this visual navigation system can run in real-time, on a single computer.

Finally, processing times involved in the outlined visual navigation system result in innovations being produced with a significant delay. Therefore, these innovations are only available once they are no longer relevant, i.e. the vehicle has moved to a new position. A minor change to the timings used by the Kalman filter can overcome this problem, allowing delayed information to be fused as normal.

The sections contained in this chapter are outlined below:

Section 8.1 presents the comparison of navigational accuracies of a number of different methodologies outlined in this thesis. The differences in performance of these systems are discussed.

Section 8.2 describes a method by which real-time operation can be achieved. This section also presents
a procedure by which delayed fusion of information can be accommodated, without affecting accuracy or computational performance.

Section 8.3 provides a summary of the outcomes presented in this chapter.

8.1 Solution Accuracy Comparison

This thesis develops a number of different visual navigation aids which can be used to constrain dead-reckoning drift. It can be seen that the use of these different aids will be dependent on the availability of different classes of a-priori information, which describe the environment. The processing power which is available on-board the platform will also merit consideration. These visual aid systems are:

1. **Spline SLAM**: Simultaneous localisation and mapping using spline features, using a DTEM to estimate the range to viewed features.

2. **Spline TAN**: Curvature based associations between viewed and mapped edge curves, and curve features stored in a pre-processed geo-referenced database.

3. **Contour TAN**: Terrain profile estimation using optical flow, and association with a known DTEM.

4. **Visual Odometry**: Vehicle velocity error estimation through the analysis of optical flow measurements, relative to inferred range to terrain from using a DTEM.

5. **DTEM Edge Estimation**: Associating viewed and mapped edge curves to estimated edge points obtained from the analysis of a DTEM.

It is pertinent to describe four discrete navigation systems, based on restrictions due to the availability of terrain information sources and available processing power. These systems and governing restrictions are outlined below:

1. Spline SLAM can be used when minimal information which describes the environment is available. This process can be assisted with the use of a DTEM, however if this is not available SLAM will still restrict inertial drift, albeit with a penalty to navigation accuracy.
2. Spline SLAM can be assisted with TAN associations if processing of an edge curve database can be performed before mission start.

3. Optical flow based terrain contour matching and visual odometry can be fused with spline SLAM, if only the DTEM is available, and the vehicle platform carries sufficient processing power. This system can also be enhanced through the use of DTEM based edge estimates.

4. If both a DTEM and edge feature database are obtainable, as well as the platform containing ample processing power, all five visual aids may be used simultaneously.

The navigation accuracy of these four systems is demonstrated in figure 8.1. This figure also shows the times at which each of the different visual aids provide an update. Although these aid timings are specifically shown for system 4 (where all aids are fused), the other methods will demonstrate similar timings. The aid timings of these alternative methods may be directly observed by reviewing the results of chapters 5-7.

Figure 8.1: Comparison of the localisation accuracies of all navigation methodologies. Also shown are the time locations of the different innovation types, for the fusion case with all described aids.

The relative accuracies of these systems shown in figure 8.1, clearly demonstrate that the spline TAN-SLAM system performance is the most accurate. A notable fraction of the TAN-SLAM navigation solution demonstrates localisation precision within 20 metres of the GPS/INS truth. It is apparent that the addition of DTEM and optical flow based aids, do improve the accuracy of spline SLAM when these
innovations are fused. However, these benefits do not appear to be very significant.

It is especially noteworthy that the addition of the DTEM and optical flow aids to the spline TAN-SLAM solution do not result in an overall improvement. In fact, the addition of this information often tends to degrade the navigation solution. This is because the accuracy of the spline TAN innovations is comparatively high, whereas the innovations from C-TAN are significantly less accurate. Therefore, the addition of this extra information tends to degrade the navigation solution. This can be seen from table 8.1, where the time-averaged position error for each system is presented.

<table>
<thead>
<tr>
<th>Spline SLAM</th>
<th>Spline TAN</th>
<th>Contour TAN</th>
<th>Visual Odometry</th>
<th>DTEM Edge Estimator</th>
<th>Mean RSS Accuracy (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>72.3</td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td>67.2</td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>34.2</td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>40.0</td>
</tr>
</tbody>
</table>

Table 8.1: Comparison of the mean RSS errors of each of the different visual navigation aids.

8.2 Real-Time Implementations

As the purpose of an aerial navigation system is to be deployed on real aerial platforms, it is important that such a system can operate in real-time. As demonstrated in chapters 5-8, given unlimited processing power, exclusively visual information can be used to localise a vehicle for navigation. However, given that real-world platforms have limited processing power, it is important to consider how these approaches can be practically implemented in real-time, while still successfully eliminating inertial drift.

Based on the results of the system comparisons in section 8.1, the spline TAN-SLAM implementation was chosen for real-time analysis.

8.2.1 Frame Skipping

All of the navigation solution results shown in this thesis, have been processed using imagery recorded at 5 Hz. As demonstrated in chapter 6, the developed spline TAN-SLAM algorithm is incapable of processing visual information within the 200 ms execution window between frames. Therefore, if every frame is to be processed, this system cannot operate in real-time.
Figure 8.2 shows how successive fusions over their time budget will result in a cumulative lag between fusion system time, and real time. This means that the present vehicle state estimate will only be known at some point in the future. As the flight progresses, this time difference will increase. Considering the speed aircraft can move, even a few seconds of delay can result in hundreds of metres of error, resulting in the position estimate essentially becoming useless.

The solution to this problem is to ignore any frames which are recorded while the system is already processing a recently captured image. Chapters 4 and 6 show that the combination of image segmentation, spline SLAM and the curvature matching TAN process, can result in upwards of 600 ms of processing for a single frame. Therefore, it is expected that there may be cases where only one out of every 3 or 4 successive frames may be used for processing, in order to perform real-time execution.

Figure 8.2 demonstrates how ignoring any frames captured during image processing, can prevent the navigation system time from diverging. This will therefore ensure that the most current state estimate is not overly delayed from the true current state.

The frame-skipping technique is utilised to perform a real-time navigation example. The computer system used for this example contains a 4.2 GHz Intel quad core i7-4770K CPU, and a 780 GTX GPU for aiding the image segmentation algorithm. Figure 8.3 shows the resulting navigation accuracy of this system, as well as the number of frames skipped in order to reach real-time.

During periods where no visually distinct edge features are available, it is clear that the system can process every second frame, i.e. only one frame is skipped per update. This is due to the time required to process each frame, however as no features result from this process, no Kalman updates or feature initialisations follow. During periods where visually distinct features are detected, the additional
processing workload of spline-based SLAM requires a second consecutive frame to be dropped, resulting in a visual update rate of 1.7 Hz. This update rate can be maintained in the presence of some lower complexity TAN updates, however more generally requires a third frame to be ignored. During this example, a small number of instances required four consecutive frames to be dropped between updates, representing an update frequency of 1 Hz.

Figure 8.3: Evolution of navigation accuracy of real-time spline TAN-SLAM. Also shown are the number of frames dropped to reach real-time.

Significant decreases in visual update frequencies are necessary to operate this system on a single computer resource, without further significant code optimisations. Figure 8.3 clearly shows that despite this, the spline TAN-SLAM aid negates inertial drift in real-time. The decrease in the number of visual sensor measurements does restrict the accuracy of the navigation solution, however figure 8.4 demonstrates that this accuracy loss is minimal. The accuracy loss can be quantified by determining the time-averaged RSS position error for both solutions. The real-time implementation exhibits a mean error of 38.8 m, demonstrating only a small average error increase over the fusion of all frames, which results in an RSS error of 34.2 m.

8.2.2 Delayed Fusion

As already shown in section 8.2, ignoring frames obtained while image processing is already being performed, can result in real-time operation of otherwise non-real-time capable algorithms. This method however, does not solve the problem of delays in the availability of new information due to processing times. Navigation, guidance and collision avoidance systems require knowledge of the present state of the
Figure 8.4: Navigation error comparison between the fusion of all image frames, versus only those possible for real-time operation.

vehicle. Delays in the derivation of new measurements will only help in determining where the vehicle was located in the past. This will result in significant problems, such as potential collisions, or unstable control systems, due to the time delay.

Fortunately, the use of an IMU can eliminate this problem. As outlined in chapter 3, the prediction step of the Kalman filter, involves the propagation of the vehicle state estimate, to any desired future time. The assumption can be made that this process will be significantly less computationally expensive, than the visual processing required for Kalman updates. Therefore, the estimated past vehicle state, containing fused visual information, can be used to propagate the vehicle state forwards to the present, using any inertial measurements collected during this time interval.

Figure 8.5 describes this process, where processing delays can be seen to result in significant lag between the true time, and the state estimate system time. This method can be made deterministic through the definition of a delayed time horizon (orange line). This horizon is deferred by an extent greater than any likely required frame processing times. This ensures that any frame innovations will be available before this delayed fusion system requires them. From this delayed fusion estimate, the Kalman prediction algorithm can be used to propagate the state estimate to the current time (pink line), using inertial measurements. Once this propagation has reached the current time, it can continue to operate on new inertial measurements, when they become available (green line). Simultaneously, whenever a visual Kalman update is made, the updated delayed state can be used to re-propagate the solution, resulting in a more precise estimate of the current vehicle state (pink line). The result of this, is that the navigation system is always estimating the current position of the vehicle, without sacrificing the accuracy gains of using computationally expensive visual information.
The application of a one-second delayed fusion with forward prediction state propagation, is shown in figure 8.6. This has been compared to the navigation accuracy of un-delayed fusion. It is clear that the use of forward predictions produces an almost identical level of accuracy to the instant-fusion assumption. The only minor drawback is that position corrections occur one second after the measurements are taken, and therefore result in a slight delay, before the vehicle state estimate converges to a more accurate solution. Despite this, the delayed fusion technique does not negatively impact the navigation solution accuracy. Furthermore, computation time is also mostly unaffected, with the re-propagation and present time predictions requiring a total of only 1.5 seconds of processing time, over the 400 second flight path sequence.

8.3 Summary

This chapter outlines the relative performance of the visual navigation aid systems developed in this thesis. It is apparent from this comparison that should edge feature information be available prior to mission start,
it is best to rely on spline based TAN-SLAM. If this information is not available, a DTEM can be used to limit drift through the use of optical flow information, as well as spline SLAM. If operating in completely unknown environments, spline SLAM can still significantly limit inertial drift, as demonstrated in chapter 5.

As the visual processing methods presented in this thesis require over 200 ms to compute, processing every frame of a 5 Hz real-time video stream is unattainable on typical desktop hardware at the time of writing. True real-time implementations would therefore require significant re-optimisations of the presented algorithms, large increases in available computational resources or implementation on a distributed processor network. Alternatively, this chapter suggests how adaptive frame-rate processing can be used to re-acquire real-time performance by ignoring images collected while still processing others. It has been shown that real-time navigation can be achieved while still eliminating inertial drift via spline TAN-SLAM, using this method.

Due to the time taken to process aerial imagery, Kalman innovations obtained from computer vision algorithms will not be valid for the current vehicle state by the time they are obtained. This chapter outlines a method by which the Kalman filter can be delayed by a small increment, thus ensuring that the processing of innovations is complete before they are required. The prediction methodology of the Kalman filter may then be used to supply a current-pose state estimate at any time, with negligible increases in computation time, and no loss of navigational accuracy.

One important consideration that must be made regarding the delayed fusion method is that the current time prediction branch must occur in parallel to the visual processing and delayed fusion state predictions. If these predictions are delayed to perform processing of other algorithms, the solution will lose accuracy, just as if this forward propagation method was not employed. This may therefore require a certain amount of the available computational resources to be dedicated to this task, potentially slowing operation of the rest of the system.
Chapter 9

Conclusions

This thesis has outlined the development and implementation of a visual navigation aid which uses monocular camera imagery to constrain drift. Video images obtained in tests are processed, with measurements fused into a probabilistic state estimator, in order to provide a robust aerial navigation solution. The use of visual imagery for aerial navigation, provides a number of significant benefits over more traditional means of navigation. Firstly, the navigation measurements are directly relative to the environment in which the vehicle operates. Secondly, on-board processing of information improves tolerance to external jamming or interference effects. This is also a result of the system not requiring any external infrastructure, such as ground or orbital radio beacons. Thirdly, the visual navigation system benefits from the use of exclusively passive sensors. These limit the power requirements of such a system, and also eliminate many mission and platform restrictions, resulting from the use of heavy, large and costly active radar systems. Passive sensors also do not affect the visibility of an aerial platform to radar detection systems, therefore aiding any stealthy operations of such a vehicle. Finally, as a computer vision navigation aid does not require a human pilot for operation, many further platform and mission restraints can be bypassed. These restrictions would otherwise be due to the physical and environmental limits a human can take, as well as the added volume and mass of humans, plus required support systems.
9.1 System Performance

This section provides a performance overview of the systems and algorithms, developed in this thesis.

9.1.1 Image Processing

The presented image segmentation and texture classifier algorithms demonstrate an accuracy of 87.5%, when determining areas of trees, grass, water and roads from aerial imagery. This value was determined through comparisons between manually designated training data and the texture estimates produced by the outlined algorithms. In the case of exclusively distinguishing between forest regions and water, this accuracy improves to 97.8%. The algorithms presented are capable of segmenting images within around 270 milliseconds (ms), with some uncommonly complex frames requiring approximately 300 ms to compute. This computational performance is based on desktop hardware of reasonable specifications (quad-core, 4.2 GHz Intel i7-4770K CPU, 780 GTX GPU), without significant code optimisations.

9.1.2 Spline Based SLAM

The image segmentation results are used to perform SLAM, using the edge points between regions of differing texture. These edge points are used to map viewed features using spline curves. The SLAM visual navigation aid is shown to be capable of significantly limiting inertial integration drift, to around 200 metres over 400 seconds of flight. This is possible without any prior knowledge of the environment. This is a significant improvement over inertial integration, which diverges to over 4 kilometres over this same time-span. If terrain height information is known in the form of a DTEM, navigation accuracy can be further improved, constraining drift to around 150 m over this 400 s flight.

The feature map resulting from the SLAM process closely represents the true edges in the environment, especially with the use of the DTEM. This SLAM process requires around 100 ms to fuse information from segmented images, with some complex frames requiring around 150 ms. These times do not include the processing required for image segmentation, which would be best performed in parallel on dedicated hardware.
9.1.3 Visual Spline-TAN

Geo-referenced aerial and satellite imagery databases are freely available and extensive. These information sources can be used to provide absolute position data, for the visual navigation aid. In order to limit the amount of information which needs to be stored on the aerial platform, aerial images can be pre-processed, using image segmentation techniques. Data compression ratios of 237:1 are shown using the algorithms presented in this thesis. The resulting geo-referenced spline feature map accurately captures high-level edge features, modelled using splines.

Data association can be performed, during flight, between the SLAM produced spline map, and the pre-processed geo-referenced edge feature database. Matching algorithms presented in this thesis use curvature based methods, to perform these associations. These resulting position innovations help to improve navigational accuracy to around 20 metres when curve matches can be made. During periods where these matches cannot be performed, inertial drift results in solution degradation. Despite this, position error is still limited to a maximum of around 100 metres using this flight data example. The spline based association algorithms presented in this thesis, can add up to 125ms to the processing time of an image, however they generally only require around half this value.

Terrain profile information provided by a DTEM can be used to adjust the geo-referenced feature database, based on changes in the environment. Water level changes have been successfully accounted for, preventing the resulting shift in water edges from biasing the navigation solution. If water level height cannot be obtained before mission commencement, in-flight associations can be used to estimate any changes. It has been demonstrated that these estimates can be extremely accurate.

9.1.4 Visual Contour-TAN

Optical flow can be used to estimate the contour profiles of viewed terrain, as demonstrated in chapter 7. This profile estimate may then be associated with a known DTEM, allowing the determination of any position drift. Furthermore, optical flow information can be used as a measurement of the vehicle velocity, providing a vision based odometry solution. This visual alternative to traditional radar-based TERCOM has been demonstrated to be capable of limiting inertial drift, exclusively using passive sensors. As this system does not require image segmentation, it does not necessitate the use of any training data.
for texture classification. This system has been shown to limit drift to around 400 metres over the 400 second flight data example.

This C-TAN system can be fused with the existing spline based SLAM navigation aid, in order to further refine the accuracy of the navigation solution. This thesis outlines a method by which the DTEM can be used to estimate the locations of water edge boundaries. Associations made between these edge estimates and the SLAM map, may be employed to help constrain navigation drift, further enhancing navigation accuracy. These associations allow absolute position innovations to be made, without the need for pre-processing aerial imagery sources. This combined system is capable of limiting drift to around 150 metres.

Optical flow based C-TAN innovations require up to around 300 ms to compute. Therefore, this method is computationally expensive, especially when fused with spline based SLAM and image segmentation techniques. Around 100 ms of this time is involved in the calculation of optical flow measurements, whereas the terrain estimation process can take around 150 ms per frame.

9.1.5 Real-Time Implementation

The solution comparison presented in chapter 8 demonstrates that the spline SLAM method, combined with curvature-based TAN, operates with the highest navigational accuracy of the various outlined systems. By varying the frame-rate of the visual processing, this system can operate in real-time, on typical desktop hardware, such as that described previously in section 9.1.1. This requires image frames to be ignored during periods when the system is already processing a new image. Real-time operation necessitates a significant number of frames to be ignored, and can involve the visual update frequency decreasing to 1 Hz, down from 5. Non-TAN spline SLAM requires less processing time, and as such can maintain a 1.7 Hz update frequency.

The reduction in visual update frequency required for real-time operation will of course amount to a significant loss of observation information. Despite this, there is no significant loss of navigational accuracy, when compared to processing every frame. Furthermore, the use of a secondary prediction branch to ensure an un-delayed navigation solution at all times, also does not affect the localisation precision. This extra processing only adds 1.5 seconds of computation, over the 400 second flight. This
represents less than a 0.4% increase in processing time, in order to eliminate any possibilities of solution degradation due to processing delays.

### 9.2 Summary of Contributions

This thesis has presented a number of new contributions and outcomes, which are summarised below:

1. Development of a visual navigation aid, which is capable of operating either with, or without, a-priori feature database information. This aid is capable of limiting inertial drift using high level, human identifiable, edge features.

2. Demonstration of a watershed-based image segmentation system, capable of determining the location of edge features present within a camera frame. Methods of improving robustness and computation speed through limiting over-segmentation have been developed. This is coupled with a system to extract these edges for use in a visual navigation aid. Texture classification assists in allowing these edges to be uniquely distinguished.

3. Development of a monocular, 6-DOF spline based SLAM algorithm, which assists navigation through the initialisation and tracking of detected edge features. This is based on the generalisation of existing 2D spline SLAM work [85, 91, 92].

4. Derivation of a spline parametric re-weighting method which eliminates the sensitivity issues caused by spline end-point clamping.

5. Generation of a data-compact, high-level feature database from existing aerial and satellite imagery sources. Texture segmentation and edge detection techniques are used to extract this information.

6. Development of data association techniques which allow matching of viewed edge features to corresponding database features. These associations can then be used to perform terrain aided navigation, allowing absolute position solutions to be obtained from visual information only.

7. Implementation of a ray-casting based range-to-terrain estimator, allowing the use of a DTEM to restrict the initialisation problems inherent to bearing-only SLAM.
8. Investigation of an optical flow based method for estimating the height profile of underlying terrain, operating exclusively using information obtained from passive visual sensors.

9. Development of a data association technique for matching optical flow based terrain profiles, with a known DTEM profile, allowing absolute position solutions to be obtained.

10. Investigation of a visual odometry implementation, which operates in parallel to the visual terrain contour matching system, limiting inertial drift of velocity estimates.

11. System validation using real flight-test data, obtained using the University of Sydney Jabiru J400 aerial test platform.

12. Development of a technique for optimising the use of visual localisation updates, which are delayed due to image processing time requirements.

### 9.3 Directions for Future Work

The work outlined in this thesis, presents an investigation into the use of high level visual feature navigation, in GNSS-denied environments. A number of areas of this work, however, could be extended and improved to provide further capability, as well as improving robustness, computational efficiency and navigational accuracy. These suggested areas for future work are outlined below:

#### 9.3.1 Image Processing

Significant improvements and optimisations, will be required in the image segmentation and texture classification algorithms. Advancements in the robustness of the segmentation process will directly benefit the accuracy of the SLAM process, refining both mapping and navigation accuracy. Optimisations to the computational implementations of these algorithms would also benefit the system, as this would allow for the use of higher frame rate video information.

Improvements in robustness and functionality of the texture segmentation algorithms, could also result in more reliable detection of other ground features. These may include roads or forest boundaries,
as well as fences. More varied detections of curve features would improve the robustness of the system, allowing successful operation over a wider scope of varied terrain categories.

Multi-spectral information could be used to benefit the reliability of the image segmentation. For example, infra-red cameras may assist in texture classification, as well as allowing operation at night, or in other low-light conditions. Extra information about the environment can also be used to improve the image segmentation process. This includes the current time of day, which may assist in the estimation of the positions of any solar reflections, in aerial imagery.

### 9.3.2 High Level Feature SLAM

A clear extension to this work would be SLAM using road features, which would automatically arise from improvements to the texture segmentation algorithms [118]. Along with this, other types of high-level features can be used for SLAM, concurrently with edge curves, such as road intersections. Previous implementations of SLAM based on many of these alternative features have already been demonstrated [60], and therefore could be used to improve the reliability and accuracy of the navigation aid.

Further optimisations of the presented spline SLAM algorithms would allow higher frame-rate video to be used for navigation. This would help to improve the navigational accuracy of the system. Lower-level features, such as SURF, could also be used to assist the navigation process. These would best be employed as temporary navigation markers, limiting inertial drift during periods where the more robust edge features are not available.

### 9.3.3 Robust Visual Spline-TAN

Improvements and extensions to both the number and types of features which can be used for SLAM, will improve navigational accuracy. The use of more features also increases the scope of potential geo-referenced feature databases which could be utilised for TAN innovations. Some possible features, such as road intersections, have already been demonstrated as TAN-SLAM aids [60]. These systems could be operated in parallel to spline SLAM, as well as any other methods, to improve the vehicle navigation
estimate accuracy. The use of multiple TAN-SLAM methodologies also enhances the robustness of the system to differing terrain variations.

9.3.4 Visual Contour-TAN

Although visual terrain profile TAN has been demonstrated to be capable of constraining inertial drift, its performance is let down by computational limits. Improving the computational efficiency of the algorithms would allow for higher frame-rate video to be processed, extending the available information for fusion. Improved computation times would also allow for higher resolution imagery to be used for the optical flow calculations, and more sampled points for non-linear visual odometry. This would also enhance the navigational accuracy of such a system.

The use of the SRTM data for the DTEM limits accuracy, as the resolution of this database is relatively low, with a 30 metre sample point spacing. This restricts the precision of profile match innovations, as well as the reliability of associations. The use of a higher resolution, alternative DTEM source, would therefore be beneficial.

A significant improvement in terrain profile mapping and navigational accuracy could be obtained through these suggestions. In this case, the terrain profile map could potentially be used to measure changes in terrain height. This capability would result in a range of useful applications, such as aerial mapping of tree growth. Such a system would be highly advantageous to the forestry industry.

9.3.5 Real-Time Implementation

Improvements in the computational efficiency of the presented visual navigation aids, would improve the performance and reliability of the visual navigation system. Shorter processing times would limit the amount of delay required for the deterministic fusion of Kalman innovations. Restricting the fusion delay will decrease the duration of inertial predictions required for the determination of the current vehicle state. This would therefore improve the accuracy and variance of the un-delayed, real-time localisation estimate. Further benefits could be obtained through the investigation of more capable computational hardware, such as the use of GPGPU systems, or FPGA for image processing. The use of a higher performance CPU
with more logic cores, and improved multi-threading of system code would also enhance performance.

As previously determined [10], SLAM functions best with the use of few, accurate and robust features. For this reason, a visual navigation aid which combines innovation fusions from multiple different feature types would benefit from a heuristic switching mechanism. Such a mechanism would be capable of determining which innovation source contains the best information at any particular moment. These innovations would then be fused into the Kalman state filter, to the exclusion of other sources. Therefore, in situations where curve features are not available, contour-TAN would be used. On the other hand, when visually distinct features are viewed, C-TAN based updates would be temporarily placed on hold.

Adding to the number of video cameras fitted and employed by the system, may also benefit the navigation system. Increasing the time over which specific features can be tracked frame-to-frame, will advance the navigation accuracy of SLAM. Expanding the field of view of these cameras would also improve the detection likelihood of robust features for SLAM, as well as features which can act as associations for TAN.
Bibliography


[10] Peter W Gibbens, Gamini MWM Dissanayake, and Hugh F Durrant-Whyte. A closed form solution to the single degree of freedom simultaneous localisation and map building (slam) problem. In


222


Chapter 10

Appendix

10.1 Parametric Curvature Derivation

Let \( x \) and \( y \) be the co-ordinates of a point \( s \) on an arbitrary curve, in 2-dimensional space. Parametric variable \( t \) defines the distance along this curve that this point lies. At point \( s \), the local partial derivatives of this curve are known, in each dimension, with respect to \( t \). i.e.

\[
x' = \frac{\partial x}{\partial t} = a \tag{10.1}
\]
\[
y' = \frac{\partial y}{\partial t} = b \tag{10.2}
\]
\[
x'' = \frac{\partial^2 x}{\partial t^2} = \frac{\partial a}{\partial t} = a' \tag{10.3}
\]
\[
y'' = \frac{\partial^2 y}{\partial t^2} = \frac{\partial b}{\partial t} = b' \tag{10.4}
\]

where \( a \) and \( b \) are derivatives of co-ordinates \( x \) and \( y \) respectively, with respect to \( t \).

It can be observed that the gradient \( G \) of the curve at parametric point \( t \) will equal:

\[
G = \tan^{-1} \left( \frac{b}{a} \right) \tag{10.5}
\]
Also, the distance this point moves along the curve due to perturbations in $t$ can be determined by:

$$\frac{\partial s}{\partial t} = \sqrt{a^2 + b^2} \tag{10.6}$$

The curvature $C$ at parametric position $t$ is equal to the rate of change of $G$, with changes in distance along the curve $s$. Using the chain rule, this can be expanded to:

$$C = \frac{\partial G}{\partial s} = \frac{\partial G}{\partial a} \frac{\partial a}{\partial t} \frac{\partial t}{\partial s} + \frac{\partial G}{\partial b} \frac{\partial b}{\partial t} \frac{\partial t}{\partial s} \tag{10.7}$$

$$= \left(\frac{-b}{a^2 + b^2}\right) a' \left(\frac{1}{\sqrt{a^2 + b^2}}\right) + \left(\frac{a}{a^2 + b^2}\right) b' \left(\frac{1}{\sqrt{a^2 + b^2}}\right) \tag{10.8}$$

and therefore, the curvature $C$ at parametric point $t$ is given as:

$$\frac{ab' - ba'}{(a^2 + b^2)^{\frac{3}{2}}} \tag{10.9}$$