Machine Vision Detection of Crop Diseases

A dissertation submitted by

Nathan Stern

in fulfilment of the requirements of

ENG4112 Research Project

towards the degree of

(Bachelor of Engineering, Mechatronics)

Submitted: October, 2014
Abstract

The production of agricultural crops such as wheat is a multi-billion dollar industry. Each year, diseases such as fungal infections can potentially destroy the entire production in a region if the conditions are right. Traditional mitigation has involved the grower’s own judgement as the first option, followed by use of trained professionals for confirmation of diagnosis and advice. Often the onset of infection is rapid, and the professionals may not always be available to assess the situation before the infection spreads. This reliance on individual judgement and off-site experts highlights a need to develop a reliable, automated software solution which can provide an accurate and immediate software diagnosis at the first sign of infection.

This research has developed into two solutions: a system to help agronomists by using professional camera equipment, as well as the outline for a mobile solution which can operate on a ‘smart’ device to provide growers with an on-hand diagnosis tool.

By investigating the way the human mind and eye works, a software emulation of the human visual system was constructed, with artificial intelligence approaches used for final interpretation of the optical response. This use of artificial intelligence has allowed for the design of a robust system which can ‘self-learn’ to recognise any new disease samples. Research involved investigation of a number of camera and hardware options.

Final system validation was conducted on both ‘stock’ disease images provided by agronomists, and on actual plant samples, which proved that the system could
function across a broad range of diseases and crops with a degree of accuracy between 95 – 99%.

This research indicates that it is possible to develop tools which can give an immediate analysis at all stages of infection, and be robust enough to work over a range of diseases and crops. Further development and refinement would provide a useful diagnosis tool for both growers and experts.
Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled “Research Project” is to contribute to the overall education within the student’s chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Dean
Faculty of Health, Engineering & Sciences
Certification of Dissertation

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Nathan Stern

Q1122382

______________________________
Signature

______________________________
Date
Acknowledgments

My sincere thanks go to my family; without their love and kindness my last years of academic study simply would not have been possible.

For dad, who’s kindness and generosity allowed me to focus my time fully on my studies.

For mum, who’s tolerance and patience helped me through each of the difficult times.

Finally for Ebony, for reminding me at 4am that it was a good time to head to bed.

I appreciate the support from my supervisor, Dr Cheryl McCarthy, for her selfless time and attention to detail during the formal writing process, and the initial introduction to the use of mechatronics and machine vision in agriculture.

Nathan Stern

University of Southern Queensland

October 2014
Contents

Abstract i

Acknowledgments v

List of Figures xiii

List of Tables xxii

Chapter 1 Introduction. 1

Chapter 2 An Overview of Symptoms of Crop Diseases. 4

2.1 Review Symptoms of Leaf Diseases. . . . . . . . . . . . . . . . . . 4

Chapter 3 Literature Review. 11

3.1 Previous Works. . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

3.2 Image Capture. . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

3.3 Image Segmentation. . . . . . . . . . . . . . . . . . . . . . . . . 14

3.4 Image Sub-Segmentation. . . . . . . . . . . . . . . . . . . . . . . 16
# CONTENTS

3.5 Gabor Filters. ................................................. 17
3.6 Artificial Neural Networks. ................................. 21

## Chapter 4 Methodology.  

4.1 Approach. ................................................. 24
  
  **4.1.1** Review of Symptoms of Diseases. .............. 24
  
  **4.1.2** Identify image capture methods. ............... 24
  
  **4.1.3** Develop suitable segmentation, and sub-segmentation techniques. ........................................... 25
  
  **4.1.4** Feature detection and Classification. .......... 26
  
  **4.1.5** Classification. ................................... 27

## Chapter 5 Image Capture Methods.  

5.1 Chapter Overview. ........................................... 29

  **5.2** Sensor Noise. ........................................ 30
    
    **5.2.1** Oversampling. .................................. 31

  **5.3** Image Compositing. ................................ 34
    
    **5.3.1** White Balance Averaging. ..................... 35
    
    **5.3.2** High Dynamic Range Imaging (HDRI). ......... 43

  **5.4** Discussion. ........................................ 48

  **5.5** Conclusions. ........................................ 48
Chapter 6  Leaf Extraction from Images. 49

6.1  Chapter Overview. ................................................. 49

6.2  Stereo Motion Capture. ........................................... 50

   6.2.1  Camera-Shake Extraction. .................................. 54

   6.2.2  Camera Movement Extraction. ................................. 63

   6.2.3  Extraction - Grown Samples. ................................. 66

   6.2.4  Stereo Extraction - Conclusions. ............................ 67

6.3  Manual Depth Extraction. ......................................... 68

6.4  Leaf Extraction - Conclusions. ................................. 71

Chapter 7  Disease Spot Extraction from Leaf Regions. 72

7.1  Disease Sub-segmentation. ....................................... 72

   7.1.1  Colour Region Masking. .................................... 73

   7.1.2  Effects of Resolution. ....................................... 76

   7.1.3  K-means Clustering. ......................................... 79

7.2  Discussion: Comparison of Methods. ............................ 83

7.3  Region Extraction. ................................................. 85

7.4  Conclusions. ......................................................... 87

Chapter 8  The Gabor Filter. 88

8.1  Chapter Overview. .................................................. 88
Contents

Chapter 11 Evaluation of the System.

11.1 Analysis of existing images. 137
11.2 Principal Components Analysis. 142
  11.2.1 Analysis of the Gabor Filter. 142
  11.2.2 Analysis of the Log-Gabor Filter. 144
  11.2.3 PCA Discussion. 146
11.3 Analysis of the Gray Level Co-occurrence Matrix. 147
  11.3.1 GLCM Discussion. 150
11.4 Colour & Texture Composite Analysis. 151
11.5 Neural Network Discussion. 153
11.6 Live Image Analysis. 155
  11.6.1 Live Image Results. 161
11.7 Live Samples Discussion. 165

Chapter 12 Design and Implementation.

12.1 System Development. 167
  12.1.1 Requirements. 168
  12.1.2 Design. 169
  12.1.3 Implementation. 171
  12.1.4 Verification. 171
12.1.5 Maintenance. ........................................... 172

12.2 Matlab Implementation. .................................. 173

12.3 Implementing on a Mobile Device. ....................... 173

Chapter 13 Conclusions and Further Work. ............................ 175

13.1 Review of Symptoms of Diseases. ............................. 176

13.2 Identify Image Capture Methods. .............................. 176

13.3 Develop Segmentation Technique. ............................. 177

13.4 Feature Detection and Classification. .......................... 178

13.5 Evaluation of the Proof-of-Concept System. ................. 179

13.6 Spectral Analysis of Leaf Diseases. ........................... 180

13.7 Implement the Machine Vision System on a Mobile Device. ... 181

13.8 Conclusions: Final System. .................................. 181

13.9 Further Work. ............................................. 182

13.9.1 Outstanding requirements: ............................... 182

13.9.2 Future Research. ....................................... 183

References. .................................................. 186

Appendix A Project Specification. ................................... 194

Appendix B Camera Operation. ..................................... 197
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B.0.1 The Colour Filter Array (CFA)</td>
<td>199</td>
</tr>
<tr>
<td>B.0.2 Camera formats</td>
<td>201</td>
</tr>
<tr>
<td>Appendix C Stereo Motion Analysis.</td>
<td>205</td>
</tr>
<tr>
<td>C.0.1 The Essential Matrix</td>
<td>209</td>
</tr>
<tr>
<td>C.0.2 The Fundamental Matrix</td>
<td>210</td>
</tr>
<tr>
<td>Appendix D Fundamentals of the Gabor Wavelet.</td>
<td>221</td>
</tr>
<tr>
<td>D.1 Gabor Filter Response</td>
<td>227</td>
</tr>
<tr>
<td>D.2 Fourier Space Conversion</td>
<td>231</td>
</tr>
<tr>
<td>Appendix E Principal Components Analysis.</td>
<td>234</td>
</tr>
<tr>
<td>E.1 PCA - The Eigenvectors Approach</td>
<td>234</td>
</tr>
<tr>
<td>E.2 PCA - The Singular Value Decomposition Approach</td>
<td>240</td>
</tr>
<tr>
<td>Appendix F Bayer Filter Extraction.</td>
<td>244</td>
</tr>
<tr>
<td>F.1 CFA Extraction &amp; HDRI - Discussion</td>
<td>252</td>
</tr>
<tr>
<td>Appendix G Matlab Pseudocode</td>
<td>256</td>
</tr>
<tr>
<td>Appendix H Mobile Device Pseudocode.</td>
<td>260</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Leaf Rust Image Samples. Courtesy of Dr Steven Neate. . . . . . 8
2.2 Tan Spot Image Samples. courtesy of Dr Steven Neate. . . . . . 9
2.3 Tan Spot Infection Stages . . . . . . . . . . . . . . . . . . . . . 10
4.1 Methodology Flowchart . . . . . . . . . . . . . . . . . . . . . . . 23
5.1 Sensor Noise Original . . . . . . . . . . . . . . . . . . . . . . . . 33
5.2 Sensor Noise Reduction stage 1 . . . . . . . . . . . . . . . . . . . 33
5.3 Sensor Noise Reduction stage 2 . . . . . . . . . . . . . . . . . . . 34
5.4 Colour Temperature. . . . . . . . . . . . . . . . . . . . . . . . . 36
5.5 Light Wavelength Intensity . . . . . . . . . . . . . . . . . . . . . 37
5.6 Colour Image, 2,800K . . . . . . . . . . . . . . . . . . . . . . . . 39
5.7 The Colour-Channel Histogram for the scene at 2,800K. . . . . 39
5.8 Colour Image, 5,200K . . . . . . . . . . . . . . . . . . . . . . . . . 40
5.9 The Histogram for the scene at 5,200K. . . . . . . . . . . . . . 40
5.10 An image shot with a manual white balance setting of 10,000K. 41
LIST OF FIGURES

5.11 The Histogram for the scene at 10,000K. .......................... 41
5.12 Colour Composite Image .............................................. 42
5.13 Colour Composite Histogram ........................................ 42
5.14 Histogram Comparison ................................................ 43
5.15 HDR 0EV ................................................................... 46
5.16 HDR -3EV .................................................................. 46
5.17 HDR +3EV .................................................................. 47
5.18 HDR Composite ............................................................ 47

6.1 Single Leaf Extraction ...................................................... 50
6.2 Stereo Vision Work-flow .................................................. 52
6.3 Stereo Vision Example Images ......................................... 52
6.4 Stereo Vision Example Reshaped ...................................... 53
6.5 Stereo Vision Depth ........................................................ 53
6.6 Left Camera Shake .......................................................... 55
6.7 Left Camera Shake .......................................................... 55
6.8 Left and Right Stereo Images - Camera Shake ..................... 56
6.9 Stereo Composite - Camera Shake ..................................... 56
6.10 Salient Corners - Camera Shake ....................................... 57
6.11 Matched Points - Camera Shake ....................................... 58
<table>
<thead>
<tr>
<th>Section</th>
<th>Figure Caption</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.12</td>
<td>Inlier Points - Camera Shake</td>
<td>59</td>
</tr>
<tr>
<td>6.13</td>
<td>Epipolar Lines - Camera Shake</td>
<td>59</td>
</tr>
<tr>
<td>6.14</td>
<td>Stereo Depth Concept</td>
<td>60</td>
</tr>
<tr>
<td>6.15</td>
<td>Cropped Composite - Camera Shake</td>
<td>61</td>
</tr>
<tr>
<td>6.16</td>
<td>Depth Map - Camera Shake</td>
<td>62</td>
</tr>
<tr>
<td>6.17</td>
<td>Left Camera Shake</td>
<td>63</td>
</tr>
<tr>
<td>6.18</td>
<td>Left and Right Stereo Images - 75mm Movement</td>
<td>63</td>
</tr>
<tr>
<td>6.19</td>
<td>Stereo Composite - Colour overlay</td>
<td>64</td>
</tr>
<tr>
<td>6.20</td>
<td>Stereo Composite - Depth Map</td>
<td>65</td>
</tr>
<tr>
<td>6.21</td>
<td>Stereo Composite - Grown Wheat Samples</td>
<td>66</td>
</tr>
<tr>
<td>6.22</td>
<td>Stereo Composite - Wheat Depth</td>
<td>67</td>
</tr>
<tr>
<td>6.23</td>
<td>Manual Extraction: Use of 18% gray reference card</td>
<td>69</td>
</tr>
<tr>
<td>6.24</td>
<td>Manual Extraction: Single leaf</td>
<td>70</td>
</tr>
<tr>
<td>6.25</td>
<td>Final extraction of single leaf</td>
<td>70</td>
</tr>
<tr>
<td>7.1</td>
<td>RGB Colour Image</td>
<td>73</td>
</tr>
<tr>
<td>7.2</td>
<td>HSV Colour Image</td>
<td>74</td>
</tr>
<tr>
<td>7.3</td>
<td>Hue Histogram</td>
<td>75</td>
</tr>
<tr>
<td>7.4</td>
<td>Histogram region Masking</td>
<td>76</td>
</tr>
<tr>
<td>7.5</td>
<td>Histogram Blurring 1</td>
<td>77</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>7.6</td>
<td>Histogram Blurring 2</td>
<td>78</td>
</tr>
<tr>
<td>7.7</td>
<td>CIELAB Colour Space</td>
<td>81</td>
</tr>
<tr>
<td>7.8</td>
<td>CIELAB Image Conversion</td>
<td>82</td>
</tr>
<tr>
<td>7.9</td>
<td>K-Means Clustering</td>
<td>83</td>
</tr>
<tr>
<td>7.10</td>
<td>Hue Histogram vs. K-means 1</td>
<td>84</td>
</tr>
<tr>
<td>7.11</td>
<td>Hue Histogram vs. K-means 2</td>
<td>84</td>
</tr>
<tr>
<td>7.12</td>
<td>Spot Region Extraction</td>
<td>86</td>
</tr>
<tr>
<td>8.1</td>
<td>Gabor Filter Bank</td>
<td>90</td>
</tr>
<tr>
<td>8.2</td>
<td>Gabor Fourier Transform Wavelet</td>
<td>91</td>
</tr>
<tr>
<td>8.3</td>
<td>Gabor Fourier Rosette</td>
<td>92</td>
</tr>
<tr>
<td>8.4</td>
<td>Gabor Spatial Filter Bank</td>
<td>93</td>
</tr>
<tr>
<td>8.5</td>
<td>Gabor Frequency (Fourier) Bank</td>
<td>93</td>
</tr>
<tr>
<td>8.6</td>
<td>Fourier Composite Rosette</td>
<td>94</td>
</tr>
<tr>
<td>8.7</td>
<td>Spatial Convolution Example</td>
<td>95</td>
</tr>
<tr>
<td>8.8</td>
<td>Rust Sample Image</td>
<td>96</td>
</tr>
<tr>
<td>8.9</td>
<td>Rust Sample Image Gabor Filter Response</td>
<td>96</td>
</tr>
<tr>
<td>8.10</td>
<td>Gabor Convolution Theorem</td>
<td>99</td>
</tr>
<tr>
<td>8.11</td>
<td>Gabor DC-Response</td>
<td>100</td>
</tr>
<tr>
<td>8.12</td>
<td>Log-Gabor Frequency Response</td>
<td>101</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>8.13</td>
<td>Log-Gabor Spatial Response</td>
<td>102</td>
</tr>
<tr>
<td>8.14</td>
<td>Log-Gabor Fourier Bank</td>
<td>103</td>
</tr>
<tr>
<td>8.15</td>
<td>Log-Gabor Fourier Rosette</td>
<td>103</td>
</tr>
<tr>
<td>9.1</td>
<td>Gabor Phase and Magnitude Responses</td>
<td>107</td>
</tr>
<tr>
<td>9.2</td>
<td>Principal Components Analysis - Leaf Example</td>
<td>109</td>
</tr>
<tr>
<td>9.3</td>
<td>Principal Components Analysis - Point Cloud</td>
<td>110</td>
</tr>
<tr>
<td>9.4</td>
<td>Principal Components Analysis - Transform</td>
<td>111</td>
</tr>
<tr>
<td>9.5</td>
<td>Principal Components Analysis - Result</td>
<td>112</td>
</tr>
<tr>
<td>9.6</td>
<td>Gabor Responses - PCA Stack</td>
<td>114</td>
</tr>
<tr>
<td>9.7</td>
<td>Gabor Responses - PCA Reshape</td>
<td>115</td>
</tr>
<tr>
<td>9.8</td>
<td>Gabor Responses - PCA values</td>
<td>115</td>
</tr>
<tr>
<td>9.9</td>
<td>Gabor Responses - PCA Images</td>
<td>116</td>
</tr>
<tr>
<td>9.10</td>
<td>Eigenspots</td>
<td>120</td>
</tr>
<tr>
<td>10.1</td>
<td>The Single Perceptron</td>
<td>127</td>
</tr>
<tr>
<td>10.2</td>
<td>The Back-propagation Perceptron</td>
<td>129</td>
</tr>
<tr>
<td>10.3</td>
<td>Artificial Neural Network</td>
<td>129</td>
</tr>
<tr>
<td>11.1</td>
<td>Methodology Flowchart</td>
<td>135</td>
</tr>
<tr>
<td>11.2</td>
<td>Spot Extraction</td>
<td>138</td>
</tr>
<tr>
<td>11.3</td>
<td>Spot Samples</td>
<td>139</td>
</tr>
</tbody>
</table>
11.4 ANN Image Structure .................. 139
11.5 ANN Image Convergence ............... 140
11.6 ANN Image Confusion Matrix .......... 141
11.7 ANN Structure Texture Analysis ...... 143
11.8 ANN PCA Gabor Confusion Matrix .... 144
11.9 ANN Log-Gabor Confusion Matrix ..... 145
11.10 ANN GLCM Magnitude Confusion Matrix .................................................. 148
11.11 ANN GLCM Phase Confusion Matrix 149
11.12 Colour + Texture ANN Structure ... 151
11.13 Colour + Texture ANN Confusion Matrix .................................................. 152
11.14 Nepenthes sample ..................... 156
11.15 Agave sample .......................... 156
11.16 Clivia sample .......................... 157
11.17 Nepenthes Colour spot extraction ... 158
11.18 Nepenthes sample ..................... 158
11.19 Agave Colour spot extraction ....... 159
11.20 Agave sample .......................... 159
11.21 Clivia Colour spot extraction ........ 160
11.22 Clivia sample .......................... 160
11.23 Nepenthes Extraction .................. 161
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.24</td>
<td>Colour composite Neural Network</td>
<td>162</td>
</tr>
<tr>
<td>11.25</td>
<td>HDR Neural Network</td>
<td>164</td>
</tr>
<tr>
<td>11.26</td>
<td>Final Design Flowchart</td>
<td>166</td>
</tr>
<tr>
<td>12.1</td>
<td>Image capture Graphical User Interface</td>
<td>170</td>
</tr>
<tr>
<td>12.2</td>
<td>Results Graphical User Interface</td>
<td>171</td>
</tr>
<tr>
<td>B.1</td>
<td>CCD sensor architecture.</td>
<td>198</td>
</tr>
<tr>
<td>B.2</td>
<td>CMOS sensor architecture.</td>
<td>198</td>
</tr>
<tr>
<td>B.3</td>
<td>Optical Sensor Construction</td>
<td>199</td>
</tr>
<tr>
<td>B.4</td>
<td>Camera Image Processing Procedure</td>
<td>200</td>
</tr>
<tr>
<td>B.5</td>
<td>Light Wavelength Sensitivity</td>
<td>202</td>
</tr>
<tr>
<td>C.1</td>
<td>Stereo Vision Cube Example</td>
<td>206</td>
</tr>
<tr>
<td>C.2</td>
<td>Stereo Vision Correspondence</td>
<td>206</td>
</tr>
<tr>
<td>C.3</td>
<td>Stereo Vision Planes</td>
<td>207</td>
</tr>
<tr>
<td>C.4</td>
<td>Stereo Vision Epipolar Points</td>
<td>208</td>
</tr>
<tr>
<td>C.5</td>
<td>Stereo Corners</td>
<td>215</td>
</tr>
<tr>
<td>D.1</td>
<td>1-Dimensional Gabor Filter Construction</td>
<td>223</td>
</tr>
<tr>
<td>D.2</td>
<td>1-Dimensional Gabor Frequency Variation</td>
<td>224</td>
</tr>
<tr>
<td>D.3</td>
<td>2-Dimensional Quadrature Gabor Filter</td>
<td>226</td>
</tr>
<tr>
<td>D.4</td>
<td>Gabor Orientation Example Octagon</td>
<td>227</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------</td>
<td>------</td>
</tr>
<tr>
<td>F.9</td>
<td>Green power curve</td>
<td>253</td>
</tr>
<tr>
<td>F.10</td>
<td>Hyperspectral CTIS</td>
<td>254</td>
</tr>
<tr>
<td>F.11</td>
<td>CTIS projection</td>
<td>255</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Diagnosing leaf diseases in wheat. ......................... 7

9.1 PCA Time Comparison ........................................ 117

11.1 Artificial Neural Network Results .......................... 153

C.1 Example Node Coordinates ................................. 215

C.2 Example Node Coordinates - Translation of Position .... 216

C.3 Example Node Coordinates - Change of Scale ............ 217
Chapter 1

Introduction.

Each year, diseases in crops such as wheat cause severe yield losses and a reduction of harvest quality.

Diagnosis of diseases in crops is a time-consuming process that involves an initial estimate of the problem by growers and their consultants, followed by further laboratory analysis by qualified agronomists. This reliance on on-site experts can result in an accurate diagnosis taking up to several weeks. Furthermore, often in remote regions the nearest expert may be far enough away as to not be familiar with local diseases, particularly seasonal mutations. As such, there is a need to provide farmers with accurate disease recognition capabilities, for example on a ‘smart’ device such as a smart-phone or tablet..

This paper seeks to outline how modern technology and computing methods can assist the grower in making the disease recognition stage simpler. Chiefly, The paper outlines steps towards developing an automated process, eliminating the need for human judgement in the diagnosis procedure, and lessening reliance on on-site facilities.

Previous works in this field such as (Kulkarni & R.K. 2012) and (Gulhane & Gurjar 2011) have focussed on using Artificial Intelligence to recognise diseases in crops such as Cotton and Pomegranate. However, these methods have focussed
solely on colour image analysis. The research presented herein seeks to develop a comprehensive image analysis process, utilising many cues from biology such as emulating the function of the eye’s cortical response to provide both colour and texture classification. The mathematical Gabor model for the eye has been used in previous intrinsic works such as fingerprint and iris identification (Daugman 2004) and handwriting analysis (Bau 2008), however the following research shows that the method can be adapted to work in the extrinsic sense, by recognising a broad range of samples from visually different diseases.

In accordance with Appendix A, the Project specification and chapters which address each of the requirements are:

1. **Review of Symptoms of Diseases.**
   A comprehensive review of Diseases prevalent in Queensland was undertaken, with the assistance of the Agronomist Dr Steven Neate, at the Department of Agriculture, Fisheries and Forestry in Toowoomba. This research is presented in chapter 2.

2. **Identify Image Capture Methods.**
   Presented in chapter 5. This chapter provides an overview of the basic camera operation, before investigating specific uses for the camera in the current scope of research. Specifically, this involves methods to improve the camera response and detection rate when using low-cost imaging devices, as well as methods for enhancing data obtained from high-quality cameras.

3. **Develop Segmentation Technique.**
   This requirement is presented in two chapters: chapter 6 presents a method for extracting a single leaf from a field of leaves, while chapter 7 investigates extracting the diseases spots themselves from the leaf, for further analysis.

4. **Feature Detection and Classification.**
   Using a mathematical model which closely approximates the human eye for image recognition is discussed in chapter 8. Correlating the data from this method is expanded on in chapter 9, while final classification using Artificial Intelligence techniques is explained in chapter 10.
5. **Evaluation of the Proof-of-Concept System.**

   The final evaluation of the complete system is discussed in chapter 11. This chapter includes both an evaluation of stock images of diseases, as well as evaluating the system on actual samples captured in the field, using the enhanced image capture techniques covered in chapter 5. The final design protocol is discussed in chapter 12.

6. **Spectral Analysis of Leaf Diseases.**

   Research into this extra requirement is covered in Appendix F.

7. **Implement the Machine Vision System on a Mobile Device.**

   The framework for developing software on a portable device such as a mobile phone or tablet is covered chapter 12.3 with associated pseudocode in Appendix H.
Chapter 2

An Overview of Symptoms of Crop Diseases.

2.1 Review Symptoms of Leaf Diseases.

Discussions with the agronomist Dr Neate from the Department of Agriculture, Fisheries and Forestry, identified that in Queensland five main fungal diseases in wheat are prevalent, with three of them grouped based on genus under the designation ‘rust’. They are:

1. Rust.

   (a) **Leaf Rust** (*Puccinia triticinia*).

   Leaf Rust is a fungal infection that exhibits reddish-orange pustules on the upper surface of the leaf only. These pustules are usually about 1.5mm across, and may be circular or slightly elliptical (DAFF 2012). These pustules actually represent the spores of the fungus at the later stages of the infection’s development. As with all rusts, the spores form in the subcutaneous layers of tissue, and then burst out of the surface upon reaching maturity.

   Depending on the genetic resistance of the plant to the particular
2.1 Review Symptoms of Leaf Diseases.

strain, the infection may be confined to a small number of sparsely located pustules on the surface, or otherwise completely cover the surface of the leaf. yellow-coloured regions may appear around the pustules; these yellow regions denote the plant’s immune system fighting the infection.

DAFF (2012) notes that most Queensland varieties of wheat have a reasonable resistance to Leaf Rust, and so Leaf Rust (as of 2012) is “not of major concern to Qld growers.”

(b) **Stem Rust** (*Puccinia graminis*).

Stem Rust may appear on both the leaves and stems, and unlike leaf rust, Stem rust may appear on the upper and lower surfaces of the leaf. Spores are usually a reddish-brown colour in oval, elongated or spindle-shaped clusters. Stem rust develops at a temperature range of 18 – 30°C and requires free moisture on the leaf such as rain, dew or irrigation droplets. Infection time can take as little as six hours, with pustules appearing 10-20 days after infection.

From DAFF (2012), Queensland wheat has a reasonable resistance to Stem Rust, however in the past stem rust has had the ability to cause 50-100% damage to crop yield when conditions have been right, and the disease has overcome the plant’s immune system.

(c) **Stripe Rust** (*Puccinia striiformis*).

Stripe Rust Also known as ‘yellow rust’, appears on the leaves and has an easily distinguishable orange-yellow colour, with small, tightly packed pustules. The infection follows the parallel venations of the monocotyledon leaf, resulting in a span-wise striped appearance on the upper surface of the leaf, leaf sheaths, awns and inside of the glumes. Stripe Rust usually develops in cooler growing conditions (10 – 15°C) and can cause up to 25% loss of yield on wheat varieties scoring a ‘moderate’ resistance rating or lower.

Two further disease found in Queensland are Tan Spot and mildew:

2. **Tan Spot** (*Pyrenophora tritici-repentis*). Tan Spot (also known as Yellow
2.1 Review Symptoms of Leaf Diseases.

Spot) presents as tan-brown flecks that develop into yellow-brown necrotic legions surrounded by yellow margins and may be 10-12mm in diameter. These surrounding yellow-brown regions may coalesce into larger regions with dark brown centres. Spots may develop on both sides of the leaves and may account for 30% yield loss in susceptible wheat varieties. Tan Spot requires temperatures of 20 – 30°C and free moisture can promote growth.

3. **Powdery Mildew** (*Blumeria graminis*). Powdery Mildew appears as white fluffy cotton-like spots. Infected regions are mostly on the undersides of leaves, however any region above ground can be infected, with infections often spreading from the underside to upper surfaces of the leaf. Advanced infections may turn grey or dark brown, with black cleistothecia forming in the mycelium mass. Whereas Rust pustules develop subcutaneously, and then erupt from the surface with obvious tissue damage around the eruption zone, Mildew develops much smaller hair-like spores which pass through the surface of the plant without eruptive damage, after which the mycelium grows on the surface of the plant.

Ideal conditions for growth involve a humid environment, and temperatures in the 16 – 21°C range. Infections may be spread by Eriosomatinae (Woolly aphids), or by spores. Mildew spores can survive between growing seasons in the detritus left after harvesting, and after infection can produce the reproductive conidium every 7-10 days. The conidia can result in secondary inoculum, and the growth and reproductive cycle can repeat throughout the growing season.
2.1 Review Symptoms of Leaf Diseases.

An overview of symptoms is given in table 2.1. A visual representation of the symptoms is given in Figures 2.1 and 2.2. Figure 2.3 shows different samples of Yellow Spot with varying degrees of infection.

Table 2.1: Diagnosing leaf diseases in wheat.

<table>
<thead>
<tr>
<th>Disease</th>
<th>Colour</th>
<th>Visual Identifier</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stripe Rust</td>
<td>Yellow/Orange</td>
<td>Small, tightly packed circular pustules, developing into stripes along leaves of older plants.</td>
<td>Upper surface of leaf, sheaths, awns and glumes.</td>
</tr>
<tr>
<td>Leaf Rust</td>
<td>Orange/Brown</td>
<td>Random, circular to elliptical shaped pustules</td>
<td>Upper surface of leaf and sheaths.</td>
</tr>
<tr>
<td>Stem Rust</td>
<td>Red/Brown</td>
<td>Circular to elliptical shaped pustules</td>
<td>Both sides of the leaf and stems.</td>
</tr>
<tr>
<td>Yellow Spot</td>
<td>Tan (Yellow Brown) necrotic region surrounded by a yellow margin</td>
<td>Spots up to 10mm, random shapes and may coalesce into larger spots.</td>
<td>Both sides of leaf, leaf sheaths stems and head.</td>
</tr>
<tr>
<td>Powdery Mildew</td>
<td>White</td>
<td>Fluffy, cotton-like appearance</td>
<td>Underside of leaf. May spread to any above ground part of the plant during advanced stages of infection.</td>
</tr>
</tbody>
</table>
Figure 2.1: Left to right: Leaf Rust, Stripe Rust, Stem Rust. Note that the middle leaf is displaying both Stripe Rust and Stem rust spores (larger, darker regions). This is due to the plant’s weakened immune system, making it susceptible to multiple diseases. This compounds the problem with machine diagnosis, as discussed in chapter 7. (Images courtesy of Dr Steven Neate.)
2.1 Review Symptoms of Leaf Diseases.

Figure 2.2: Left to right: Tan (yellow) spot. Mildew. (Images courtesy of Dr Steven Neate.)
2.1 Review Symptoms of Leaf Diseases.

Figure 2.3: All five leaves in this figure are infected with Tan Spot. This shows how the disease can vary visually, depending on the plant’s immunity to that particular strain of infection. This image illustrates one of the difficulties associated with machine vision for this task; how to make the software recognise all five leaves represent the same disease. One approach is to present many examples of leaves with different severities to the training network, covered in Chapter 10 and program it to group them all under the same disease heading. (Images courtesy of Dr Steven Neate.)

As the three rusts are visually very similar, and usually require tissue cultures to discern which strain is present, the decision was made to focus simply on separating Mildew, Tan spot, and Stem Rust with software.
Chapter 3

Literature Review.

3.1 Previous Works.

Initial research into disease detection in plants yielded a number of different approaches, with several papers available to discuss the different methods. Most prevalent among these was the use of Artificial Neural Networks (ANN), and various colour and texture feature extraction techniques to aide in classification.

(Huang 2007) used ANN and mean-grey level in R,G,B colour space to detect diseases in Phalaenopsis spp. Diseases analysed were bacterial soft rot (BSR), bacterial brown spot (BBS), and Phytophthora black rot (PBR). In this approach, the diseased lesions on the plant were initially isolated using an adjustable exponential transform. After this segmentation stage, Gray-level Co-occurrence Matrix (GLCM) methods were employed on the extracted spots in order to build up a feature vector unique to each disease spot. ANN was then used as the final classifier. Huang (2007) claimed $\approx 90\%$ accuracy in classification accuracy across the three diseases, and 97$\%$ detection accuracy in BBS.

Dhaygude & P.Kumbhar (2013) proposed a different approach for detecting diseases in Grapefruit: The image was first translated into a HSI colour space, as it was claimed that HSI gives a better colour descriptor. The green (healthy)
regions of the leaf were then masked out to leave only the diseased regions, which were then grouped into $32 \times 32$ pixel patches, and colour co-occurrence methods applied to produce colour and texture features. The results of these features were not analysed further, however the author noted that ongoing work would investigate ANN in order to increase the recognition rate of the classification process. Gulhane & Gurjar (2011) also used colour image segmentation and ANN, in detecting diseases in cotton. Arivazhagan, Shebiah, Ananthi & Varthini (2011) used a similar approach, and expanded the approach to cover 10 different plants. Associated diseases accuracy with this approach was claimed to be as high as 94%.

Jaware, Badgujar & Patil (2012) used K-means clustering for segmentation of disease spots, and found it to be a general solution to finding disease clusters in datasets. However it was noted to be computationally expensive.

Kulkarni & R.K. (2012) used a process of CIELAB colour space, combined with Gabor filters for feature extraction, and finally ANN for classification for disease in pomegranate. Accuracy using this method was claimed to be between 83% and 91%, and was deemed high to warrant further investigation into Gabor filters. However, despite this the cited article gave little in the way of verification: the Gabor filter equations seemed to be incomplete and only the results of the ANN stage were presented.

A more detailed review of the literature on Gabor filters is given below in its own section.

### 3.2 Image Capture.

A review of imaging device operation was conducted, with Sensorcleaning.com (2010) giving a good description of the differences between camera sensor types, and the specific operation of both CMOS and CCD sensor designs. The basic difference between the two was explained that CCD sensor design is cheaper to
3.2 Image Capture.

manufacture, but the photos produced are generally of lower quality. CMOS is generally reserved for higher quality camera designs.

This basic description of camera operation was expanded by (Rowse 2014). After the sensor has captured the data, it may be stored either as RAW file format, or as JPEG. The latter file format is very common, but uses lossy data compression as well as proprietary algorithms to enhance the image to appeal visually to the human eye. This compression and enhancement necessarily changes the data as initially captured by the camera sensor, and is destructive: the original sensor data cannot be recovered after the JPEG transformation has been completed. By comparison, many high end cameras such as digital SLR cameras are able to directly save the sensor data directly as RAW file format. RAW format usually requires manual post processing by the photographer (analogous to developing traditional film). However RAW format offers a number of possibilities in machine vision, such as directly extracting each of the image colour channels.

(Lodriguss 2013) explains a technique used to over-sample image data, which increases the noise-to-gain ratio of the final image. This process is common in astronomy and microscopy photography fields, where the process can be used to enhance subtle detail in the image, such as faint stars which would be otherwise obscured by camera sensor noise.

Elder (2004) explains that the human eye is capable of capturing a much wider gamut of illumination energy than the camera sensor can. This means that if a scene has a very large dynamic range (i.e., contrast between light and dark in the same scene) then a camera will typically either under-expose or over-expose parts of the scene, which exceed the sensor threshold. Elder (2004) explains that by taking a number of photographs with varying luminance gain, then combining them into a single image, that the camera limitation to high-contrast scenes can be overcome. This process has several benefits to machine vision, as it effectively allows for a much larger energy gamut to be recorded, which no single image alone would capture. This ability to record the absolute energy from each part of the scene is of benefit to machine vision, and is investigated accordingly in the
3.3 Image Segmentation.

Initial research into segmentation methods suggested that a stereo vision approach may be suitable. Traditional stereo approaches fall within two main categories (Davies 2012): either a pair of cameras in a fixed ‘binocular’ mount arrangement, or using multiple cameras in a free (mount-less) arrangement. A variation of the latter option is to use one single camera which is free to move between positions. It was decided to investigate this third option, using the motion inherent with holding a camera by hand and taking two photos in quick succession, roughly 1/2 second apart.

Investigation into the algorithms associated with reconstructing a depth map from stereo imagery involved reviewing a number of sources: (Davies 2012), (Pollefeys 2002), (Hartley 1997), (Hartley & Zisserman 2000) (Chojnacki, Brooks, Hengel & Gawley 2003), (UW CSE vision faculty 2009).

Davies (2012) highlighted the basic principals behind stereo vision, as well as briefly covered Stereo from Motion. Covered in this reference was the fundamental differences associated with using fixed vs. free cameras in stereo vision: in the former situation the distance and angles between cameras (and hence photographs) is known, and in most cases depth for a point such as a leaf which is present in both images can be found by simple trigonometry.

By comparison to a fixed binocular arrangement, when the cameras are free to move this adds another seven degrees of freedom\(^1\) between images. As the only indicator as to how much the camera is moved is contained within the images themselves, it is thus necessary to use the photos to first reconstruct the positions of the cameras at the time of image capture, and then use this to ‘calibrate’ the

\(^1\)Three translational, three rotational, one in zoom. Lens effects and depth of field are usually included in the latter.
3.3 Image Segmentation.

images so that the corresponding depth points can be extracted.

Whilst Davies (2012) explains the basic process well, the actual implementation of the algorithms associated with depth extraction are only covered in a very cursory sense. Hartley (1997) On the other hand, covers the algorithms in depth, but doesn’t elaborate on their usage.

From (Hartley 1997), the principals in building up the program could be developed: the independent seven degrees of freedom are augmented with at least an eighth\(^2\). The degrees of freedom are thus solved using matrix notation, by taking a corresponding number\(^3\) of points from the frames.

In Hartley & Zisserman (2000), the eight-point algorithm is elaborated on and its reliability proven by making it less sensitive to noise in the original images, while Chojnacki et al. (2003) also seeks to improve the performance of the original algorithm by normalizing the data in the first steps, giving better conditionality of the fundamental matrix.

Once the steps required to ‘calibrate’ images are completed, it is possible to work out the exact geometric location of both cameras, as well as the objects in the scene just from the two images. This effectively brings the ‘free’ camera problem back on par with the fixed binocular camera method, allowing for trigonometric locations of the points in the scenes to be computed. This trigonometric step is elaborated on by UW CSE vision faculty (2009), which offers several alternative methods for actually finding the points in the scenes with the least amount of error.

\(^2\)(Hartley 1997) identifies this eighth degree of freedom as containing the ‘noise’ present in the photos.

\(^3\)A minimum of eight points is required to compute the eight degrees of freedom, however often many more points are chosen. This permits ‘redundancy’ in the solution.
3.4 Image Sub-Segmentation.

Once a single leaf has been extracted from the scene, it is necessary to extract the individual disease spots from the leaves. Initial research into this process suggested it would be possible to simply pick the green healthy regions of the leaves and remove them, leaving only the infected spots behind. As ‘green’ as the camera records it is dependent on a number of factors such as lighting, as well as the overall health of the plant, there needs to be some way to discern from the scene specifically what ‘green’ is.

Rosenfeld & de la Torre (1983) discusses a method to adaptively analyse the colour threshold of the image, by first converting the colour image into an appropriate grey-scale colour space, and then plotting the histogram of the image. As the histogram reveals the absolute count of each pixel colour (or in this case grey value) then a large number of green pixels will appear as one peak on the histogram, with a small concave local minima between the green regions and the next highest colour, which is likely the colours associated with the disease itself. By locating this local minima Rosenfeld & de la Torre (1983) proposes that the colour regions of interest can be directly extracted.

An alternative method of grouping the spots is K-means clustering, as proposed by (Davies 2012). This method uses an iterative sampling method to group the pixels in the image according to which of their neighbours they best match. In an ideal sense this algorithm would group all green or near-green parts of the leaf together, and all disease spots together as another. One issue with K-means however is it is very sensitive to initial conditions: if both initial cluster estimates are in the same region then the software will attempt to extract green from green, and likely ignore the disease regions, grouping them into whichever green cluster they best fit. It is this sensitivity to initial conditions which is discussed in (Tan, Steinbach, Kumar & Ghosh 2006), where suggested improvements to the basic algorithm include running the software multiple times to improve the detection rate, and bisecting the initial clusters. This method is expanded in (Savaresi & Boley 2001), where it is explained that the centroid of the initial cluster is
chosen and half of the clustered region bisected into two sub-clusters. Analysis of these sub-clusters indicates if the initial clustering is accurate; if it is then both sub-clusters will be similar.

MathWorks, Inc. The (2010) discusses a different method of improving the initial cluster performance, by instead randomly sampling the image and taking a sub-sample of 10% of the original image. This much smaller dataset is then used to establish the initial clustering conditions.

CIELAB colour space is explained in (Adobe Systems Incorporated 2000) and expanded in (MathWorks, Inc. The 2010). This colour space closely represents the colour perception of the human eye, both in the way colour is perceived, and in the absolute response curves of human vision. By combining this colour space with K-means, it is possible to accurately cluster and extract the disease spots in the same way the human eye perceives the damage on the leaf.

### 3.5 Gabor Filters.

The creation of the Gabor filter is credited to Dennis Gabor, who first explained the process as early as 1944 in his article “Theory of Communication” (Gabor 1946). However it wasn’t until the 1980’s when the process began to be applied to image analysis, with most work in this field attributed to Dr. John Daugman (Daugman 1985). Daugman (1985) discovered that the Gabor filter closely models the optical cells in the visual cortex of mammalian brains, while experiments by Jones & Palmer (1987) discovered that the real component of the complex filter is a close match to the striate cortex in the feline eye. This realisation sparked a lot of interest in the mid 1980’s into using the Gabor filter in machine vision applications: the logic being that the ‘organic’ basis of the filter would permit the machine to detect the same visual cues in a scene as a human can.

Interest in the biological, as well as mathematical implications of this approach
makes it attractive to a number of machine vision tasks. Daugman (2004) has successfully used the method in iris detection of individuals. In this application the iris of the eye is seen as a unique biometric identifier. Commercial applications of Daugman’s research have been installed in airport security as an alternative to fingerprint identification.

Other applications have used the filter as a fingerprint identifier (Munir & Dr. M. Y. Javed 2014), as a face detector (Bellakhdhar, Loukil & ABID 2013), (Bouzalmat, Belghini, Zarghili, Kharroubi & Majda 2011), for text digitization using Object Character Recognition (Bau 2008) and for crop disease detection in cotton (Hrishikesh, Kanjalkar & Prof. S. S. Lokhande 2013).

Despite the wide ranging usages of the Gabor filter however, there seems to be little correlation as to how to apply it. Most sources (for example (Hrishikesh et al. 2013)) either generalise the steps required to produce meaningful results from the filter responses, or otherwise skip this step entirely.

For the filter to work, it needs to be applied in multiple frequencies and orientations. As explained in chapter 8, it is customary to use 8 variations in filter orientation, and 5 variations in frequency, giving a filter bank of 40 wavelets. However, this multiple application produces large amounts of data for each image: if a single spot on a leaf was $16 \times 16$ pixels, then the filter response would be $16 \times 16 \times 40 = 10,240$ pixels. A diseased leaf may have upwards of a hundred separate spots on it, and so very quickly the amount of data generated by the Gabor filter becomes unmanageable without data-compression techniques. Research into this field indicated these correlation steps are non-trivial. To date, most of the research into Gabor filters has been to identify these steps.

Daugman (2004), explained that as the filter is numerically complex, the magnitude of the real and complex components contains the information related to white balance and illumination in the original image. Similarly the phase correlation between real and complex contains the frequency/texture component. Daugman claims that by simply taking the phase component and quantizing

\footnote{It has real and complex components, existing in a 4-Dimensional space.}
it into 90° quadrants, that the filter response becomes invariant to differences in white balance and illumination. In other words, a picture of a leaf on a dark cloudy day will appear to have an identical phase response as one taken in full daylight, with the magnitude component of the filter instead containing the differences in illumination.

The approach of using phase-based frequency analysis could be useful for the application at hand, as any images captured of diseased leaves would give the same filter response regardless of illumination. However, after personal correspondence with Professor Daugman the advice given was: “the phase-based use of such wavelets for iris recognition (identification) is probably not relevant” to the disease identification task. This method however will be investigated in due course, to decide if it can be made relevant.

As a possible alternative to using a pure Gabor filter, Cook, McCool, Chandran & Sridharan (2006) outlines the use of a log-Gabor filter by building the filter itself using a logarithmic component in Fourier space and then applying this in the conventional manner. Cook et al. (2006) claims that this approach also gives an illumination-invariant response. If this method can be validated, then any of the data extraction methods outlined here would be similarly improved without relying simply on the phase response of the standard Gabor filter.

For producing meaningful results from the filter responses, Bellakhdhar et al. (2013) suggests using Principal Component Analysis (PCA) to reduce the size of the vector (data) produced by the Gabor filter. In this approach, each of the $n \times n$ filter responses are reshaped into an $(n)^2 \times 1$ vector. i.e., if the original filter response was a 16 pixel x 16 pixel image, it would be reshaped into a 256 x 1 vector. By repeating this procedure for each discrete filter response and then combining all 40 into a 256 x 40 matrix, it is possible to then take eigenvectors and eigenvalues to ‘rate’ the individual pixel correspondences. This then allows the least significant correspondences to be discarded, reducing the size of the feature vector without significant data loss.

---

5Via email discussion on 26-April-2014.
3.5 Gabor Filters.

The PCA approach seems to have a lot of merit and is straightforward to understand, however Bouzalmat et al. (2011) indicates that PCA is computationally expensive, and suggests an alternative approach: instead of taking the eigenvector approach of PCA, it is suggested instead to rely on Random Projection (RP). Random projection apparently offers better data compression and dimensionality reduction over PCA, when large vectors are concerned\(^6\). RP differs from PCA in that the latter relies on mathematical principals (eigenvectors), while Random Projection relies on statistical analysis to reduce the size of the data. Proponents of RP indicate that with high-dimensional data, error loss is reduced while at the same time providing greater compression, and with much reduced computational time.

However, while the basic logic behind the statistical approach is sound, the proofs associated with this method rely on complex lemmas and seemingly empirical validation.

A different approach is suggested by Bau (2008) for handwriting recognition, whereby instead of relying just on the Gabor filter response, some of the original data from the image itself is fed back into the feature vector. Each of the 40 filter responses is treated as a single object, and the energy contained each is extracted using texture measures such as the Grey Level Co-occurrence Matrix (GLCM) approach. This energy method feature vector is then augmented with parts of the original image, allowing for colour and spatial features to be included.

In conclusion, the broad use of Gabor filters in machine vision applications is well established, however the wide array of methods to implement them means that a careful comparison is required before deciding on the best approach for the current project.

---

\(^6\)Refer Bingham & Mannila (2005) for a comparison between Random Projection and PCA.
3.6 Artificial Neural Networks.

From (Burger 1996), Artificial Neural Networks are processes which are modelled on the neural structure in the brain, but on a much smaller scale. Neural Networks are a common tool in artificial intelligence.

ANN contrasts with traditional computer architecture in that conventional computing uses deterministic steps to achieve a solution: software operation is sequential and logical, and the transition of data through the software can be easily traced between one operation (line of software code) to the next. By comparison, ANN architecture is sparse: it uses a great number of very simple processors which apply a weighted sum to the input data. As such the entire system responds simultaneously in parallel to the inputs. Information between processing steps is not stored in variables as with a serial computer, but rather the ‘state’ of each neuron represents the ‘knowledge’ of the complete network.

By requesting a known output for an input dataset, the network can self-train by adjusting the neuron weights until the computed output matches the desired output. Neural Networks therefore are particularly useful at finding patterns in data, particularly when the diversity or amount of data is great. It is this dynamic, non-linear training process which is of particular benefit to the disease recognition process, since the pattern-finding process can be used to recognise which disease a sample spot belongs to, once the network has been trained on a known dataset.

Burger (1996) gives a very thorough overview as to what an ANN is, but leaves the finer points of their construction to other sources. Davies (2012) expands on the ANN construction process, and explains the importance using more than one interconnected hidden neuron layer so that the software stands a better chance of recognizing subtle patterns in more complex datasets by implementing more complex decision boundaries during training. Heaton (2008) expands on this information further, by defining guidelines for the number of neurons contained within each hidden layer.
Chapter 4

Methodology.

Initial research into the project objectives yielded multiple alternative approaches for each one. The decision was made to attempt to take a ‘compartmentalised’ approach, i.e., to treat each objective as a separate entity with interconnected results. This approach allowed each objective to be analysed in isolation from the others, using (initially) a manual data-set for each and ultimately replacing with automated results from the preceding sections. Figure 4.1 shows the concept flowchart, outlining each of the alternative approaches for each section. Each approach is individually explored, and the best results from each chosen for the complete system.
Figure 4.1: The flowchart shows the individual alternate processes for each stage, which will be analysed in each chapter.
4.1 Approach.

The chosen approaches of investigation for each of the sections is listed below. Although most chapters use existing image and signals processing methods to some extent, the scope of innovation for each section is also defined.

4.1.1 Review of Symptoms of Diseases.

Research began by investigating disease symptoms and visual identifiers in accordance with item 1 in Appendix A. This research and findings is presented in Chapter 2. In Queensland there are five fungal infections which are most prevalent; the decision was made to focus on separating three of these to validate the final software solution.

4.1.2 Identify image capture methods.

This section focussed on a comparison of available image capture devices, and what can be achieved using ‘simple photography’, or more elaborate imaging techniques. Devices considered for analysis were:

1. Use of a Digital Single Lens Reflex (SLR) camera.

2. Use of a digital compact camera, or camera embedded in ‘smart’ device.

Research into image capture methods is listed in Chapter 5, and is expanded in Appendix F.

Scope of Innovation.

One of the issues with image capture on compact devices is the quality of the photographs taken. As later chapters use texture analysis as a classifier, it is
important to minimise sensor noise from low quality cameras. The method chosen for noise reduction is image over-sampling. While this over-sampling method is used in other fields such as astro-photography and microscopy, the usage in this chapter seems to be the first time the method is applied to machine vision. Furthermore, this over-sampling method is expanded in chapter 5 to include colour balance corrections, which is an entirely novel approach. This chapter also mentions the use of High Dynamic Range photography, a method typically used for creating artistic images, however the HDR process allows a much greater gamut of data to be recorded from the camera sensor. Using this method in machine vision is also an original approach.

4.1.3 Develop suitable segmentation, and sub-segmentation techniques.

Initially, a single leaf needs to be extracted from the surrounding scene for analysis, with a further sub-segmentation required to extract just the diseased regions on a single leaf. Research into the initial-stage segmentation involved the use of stereo vision methods, and is explained in chapter 6.

Sub-segmentation of the diseased leaf involved methods to extract the individual disease spots from the healthy regions of the leaf. As there is a large variation of appearance in the disease depending on the severity and age of the infection (see chapter 2), it was quickly realised that simple segmentation and subsequent classification based on colour alone was inadequate. Thus, methods for sub-segmentation such as K-means Clustering and Colour Region Masking were investigated as discussed in chapter 7. This chapter also gives an analysis of the effects of camera quality on the disease extraction techniques.
4.1 Approach.

Scope of Innovation.

During the secondary sub-segmentation stage, a thorough comparison between histogram extraction vs. k-means clustering was carried out. In both cases, the camera sensor quality was again considered, with both histogram and K-means methods being trialled at increasingly lower resolutions, to indicate whether camera quality affected the detection rate of the individual disease spots.

4.1.4 Feature detection and Classification.

This section focussed on ‘grouping’ samples of known diseases based on shape, colour and texture (frequency analysis) in such a way that subtle variations in appearance between two spots on a leaf can still be grouped as the same disease in the internal data-bank.

Once the diseases are visually grouped, there needs to be some method of cross-comparison, so that when an unknown (sample) image is presented to the software, it has some way to compare and identify which disease in the data-bank provides the closest match. Research revealed use of the Gabor filter as a useful tool in object recognition. This filter approach and subsequent variations are discussed in chapter 8. The Gabor filter produces large amounts of data for each disease spot requiring an interim data correlation step before classification. This is explained in chapter 9.

Scope of Innovation.

Gabor filters have been traditionally used for recognising objects such as handwriting, fingerprints, and personal identification from the iris: the filter is used for recognising samples of a similar nature, rather than identifying different data-sets (such as unrelated infections). From discussions with Dr John Daugman\(^1\) the advice given was that while extra steps in analysis would be necessary to diagnose

\(^1\)Via email, on 24-April-2014.
samples across multiple diseases, that the specific phase-based use of Gabor wavelets as used in the iris-recognition approach was “not relevant” to the disease recognition topic. However, as phase is a valid classifier in signals processing, it was decided to pursue this line of research. By further investigating the Gabor process, the research contained in the chapters 8 and 9 has shown the use of Gabor filters to be a viable method for both grouping diseases for each infection, and also identifying if a spot belongs to an unrelated data-set. By applying signals analysis methods, the research presented here allays Dr Daugman’s concerns by proving the phase response to be a viable recognition method for both grouping disease spots, and recognising disparate infections.

To date, all of the literature on Gabor filters has involved filtering in the spatial domain. The research in chapter 8 shows significant improvements in processing time by filtering in the frequency (Fourier) domain. While transforming to a new domain is common practice in signals processing and control systems, using this approach has not been explicitly reported as an image analysis technique; experimentation into using Fourier space has resulted in reducing the processing time by a factor of 10.

By its nature, the Gabor filter necessarily produces a large amount of data for each disease spot. From the various literature reviews conducted, there is very little discussion on how to correlate this data. Chapter 9 discusses three different methods for extracting usable data from the Gabor filter response: while each of these are standard data-mining techniques, the research on applying these to the Gabor data-set is original.

4.1.5 Classification.

The Gabor filter approach closely emulates the function of the human eye. In order to correctly discern the separate diseases, an artificial intelligence approach was implemented. This method uses Artificial Neural Networks to self-train on the disease samples. The full approach is explained chapter 10 and expanded in
4.1 Approach.

Chapter 11.

Scope of Innovation.

The concept of an Artificial Neural Network is well established. However each network needs to be constructed on a ‘per-use’ basis. Chapter 11 highlights the construction of the network, and adjusting the structure to give the best disease recognition results.
Chapter 5

Image Capture Methods.

5.1 Chapter Overview.

Quoting from Davies (2012): “In vision, everything depends on image acquisition.” Given the importance of image capturing, this chapter and accompanying Appendices B and F are used to cover the wide array of imaging devices and cameras available both as dedicated vision devices, and consumer equipment ‘repurposed’ for machine vision applications. As any real device experiences sensor noise to varying degrees\(^1\), this chapter starts with a discussion on what can be done to mitigate the noise present in low-quality sensors, before presenting two innovative approaches to improve the camera colour and exposure gamut.

The wide variety of devices and capture methods available necessitated a review and selection of which devices would be appropriate for this project, as the large number of embedded devices alone that the growers may have access to means that any design needs to be ‘robust’ enough to allow it to work on cameras with differing characteristics and capture quality. Furthermore, investigation into the benefits of using higher quality Single Lens Reflex (SLR) cameras by agronomists and other specialists was investigated.

\(^1\)Noise in this regard is considered as stray pixel responses which weren’t part of the original scene. Usually low quality devices exhibit more noise.
This chapter and companion appendices focus on a comparison of what can be achieved using ‘simple photography’, or more elaborate imaging augmentation techniques, using:

- A Single Lens Reflex (SLR) camera. A 13 Megapixel (MP) Canon 5D was available for the experiments covered in this chapter.

- A digital compact camera. A Kodak EasyShare Z740 5MP was used.

- A camera embedded in a ‘smart’ device. Experiments were conducted with a Samsung Galaxy Tab 3, using the integral 3MP camera.

Appendix B provides a greater overview of camera sensor operation, camera sensitivity, and file formats, as most professional cameras can shoot in both RAW and JPEG formats. The research implications of sensor operation is discussed at the end of this chapter, and is also expanded on separately in appendix F.

Sensor operation was explained in (Sensorcleaning.com 2010). Photographic formats were explained in (Rowse 2014). Augmentation techniques such as oversampling for noise reduction were covered in (Lodriguss 2013).

Investigation into augmentation has lead to several novel applications discussed below: White Balance Averaging has been successfully applied to increase cross-channel colour correlation, while applying High Dynamic Range and Noise Reduction methods to machine vision applications is seemingly innovative.

## 5.2 Sensor Noise.

Appendix B explains that while higher-quality cameras can shoot in RAW format, that most lower-end cameras can only save the data using the lossy-compression JPEG algorithms which also interpolate and compress the captured data. This coupled with the construction of the CCD sensor results in noise in the recorded image. The amount of noise usually varies between devices and sensor quality,
and can also vary due to conditions at the time of capture, such as temperature
variations and other sources of interference.

As the image recognition stages discussed in Chapter 8 rely on frequency analysis
for accurate diagnosis, it is desirable to limit the effects of noise in the image
as much as possible during the image capture stages. The following section
discusses possible methods of noise reduction. It should be noted that depending
on the quality of the camera, noise reduction may not be necessary, however the
subsequent sections present general photographic improvement methods. The full
benefit of using these methods will be discussed in chapter 11.

5.2.1 Oversampling.

Noise in an image can be attributed to two main causes; static noise is a function
of the camera itself and may depend on sensor construction (e.g., light leakage
from one pixel to the next), and dynamic noise such as sensitivity gain with
temperature. Often there is some degree of overlap in the definitions. According
to Fellers & Davidson (2004): noise can be thought of as either ‘temporal’, i.e.,
varying with time, or ‘spatial’, in that the same noise spot will appear consistently
across multiple images. While most sensors produced are designed to be ‘noise
free’, this however means they exhibit Gaussian noise distribution with a zero-
mean. The end effect is that two images of the same scene taken sequentially
may exhibit random variations in temporal noise.

Whilst spatial noise is a function of the camera and thus can be removed on a
per-device basis with software filtering, temporal noise is usually harder to remove
using such approaches. However, the temporal variation in noise itself allows for
it’s removal from the image, since whilst the noise is random the actual signal
remains constant. By taking multiple exposures of the same scene and merging
them into a composite image, each pixel becomes over-sampled, allowing for the
signal noise to gain (SNG) ratio to be improved. This over-sampling technique is
common in microscopy, and astro-photography fields (Lodriguss 2013). However,
usage of this technique in machine vision as presented here has not been explicitly reported in the literature, and is thus considered an innovative approach.

Compositing works because the value of each final pixel is the mean of the sum of each of the contributing images. If the SNR of a single image is considered as 1, the compositing will increase the SNR gain by:

\[
SNR_{Gain} = \frac{n}{\sqrt{n}}
\]

Where \( n \) is the number of images.

An example of this technique is shown in figures 5.1, 5.2, 5.3. With two composited images, the SNR gain is 1.414, while with 12 images, the gain is 3.46 times that of a single photograph. By comparison with other noise reduction methods such as Gaussian or low-pass filtering, image stacking does not result in a loss of fine-resolution detail (Davies 2012). The following images were constructed by the following process:

1. The individual images were recorded using the device and saved separately.

2. Each pixel value across the image was divided\(^2\) by the total number of images \( n \).

3. The images were merged by adding each together on a per-pixel basis.

\(^2\)It was found that adding the pixels at full strength results in clipping, if the total pixel value exceeds the 8-bit colour range. Thus it was necessary to divide before adding, rather than adding and then dividing to compute the mean pixel value.
5.2 Sensor Noise.

Figure 5.1: Original image, cropped from a 3 megapixel image taken with a Samsung Galaxy Tab 3. Note the noise in the image, particularly in the blue sky region, which should appear to be a solid blue without any texture or ‘grain’.

Figure 5.2: Stack of 2 images Galaxy Tab 3. Sensor noise has been reduced by 1.414.
The preceding section covered the effects of noise in an image, and explained how over-sampling each pixel with multiple images could be used for noise reduction. This section will discuss further benefits from compositing multiple images. Since sensor noise itself is independent of the other functions of the camera at the time of the exposure, rather than using multiple images shot with the same camera settings it is possible to reap the benefits of SNR gain while also improving the image in other ways. Of the two methods discussed below, White Balance Averaging is completely innovative, while applying High Dynamic Range Imaging to machine vision also appears to be novel.
5.3 Image Compositing.

5.3.1 White Balance Averaging.

The same scene shot under different lighting will produce different responses of the camera sensor (also known as colour-cast). For example, an image shot late in the afternoon will result in a higher red level due to atmospheric scattering of the blue light, compared to an image taken at midday which will appear to have a bluish tint.

Under most instances, the camera is able to correct this to a degree. Often the camera samples a spot in the middle of the image (center-weighted) and compares the histogram across the three colour channels to what the processor considers to be the ideal. At the JPEG stage the histogram is then reshaped to match the ideal curves (Canon 2013).

While this automatic white balance correction is usually sufficient for outdoor images taken at midday, the camera usually can’t cope with ‘tricky’ lighting situations, such as under artificial lighting, or on overcast days. From (Canon 2013), the camera sensor simply assumes that every object in the image reflects 18% of the light, and the camera exposes accordingly, regardless of the actual objects reflectance\(^3\).

One solution to this is to expose the image with a known white reference\(^4\), either within the image itself, or more often taken as a separate image under the same lighting. By giving the processor a surface with known white/gray values, it is able to adjust the lighting more accurately. In professional photography circles however, lighting is manually adjusted, either as a ‘best guess’ based on experience and knowledge about the lighting conditions, or by using a dedicated light meter.

In manual white balance correcting, the colour temperature of the image is considered. Under certain lighting conditions, a specific colour temperature will be apparent. Any subsequent photos shot under this same light will then have the

---

\(^3\)Snow, and glossy leaves for instance may have a much higher reflectance balance, while dark spots on a plant may be lower.

\(^4\)White or usually 18% grey cards are standard tools in digital and film photography.
same colour temperature and thus same white balance. This colour temperature is a direct comparison to black-body radiation temperature, and is expressed in Kelvin. Refer figures 5.4 and 5.5.

Figure 5.4: Colour temperature in degrees Kelvin. A hotter temperature results in a shift towards the blue spectrum. Image adapted from (Photography Mad 2014).
Figure 5.5: Shows the relative intensity (energy) of light under colour temperature. Image adapted from (Photography Tricks 2014).

Figure 5.5 shows the energy of light under different colour casts. By taking multiple exposures of the same scene under a different camera colour temperature setting and then compositing them, the original white balance data can to some extent be restored across the histogram. This process restores some of the data lost to JPEG histogram shaping within the camera processor.

Results of experiments with a canon 5D mk1, taken over a colour range of 2,800K to 10,000K, and the subsequent composite image are shown below in figures 5.6 to 5.13. A final side by side comparison of the image histograms is visible in figure 5.14. In the final Composite histogram, it is evident that averaging the values across all three images has resulted in a very tight distribution between all three colour tones, indicating an improvement to the white balance and restoration of the colour energy (i.e., the area under each curve) which was otherwise discarded from the original JPEG images during the camera’s internal white balance correction phase at each setting. Note in each case the left-
weighting in the histograms. This indicates a higher number of darker pixels in each image. This too can be corrected using High Dynamic Range compositing techniques, which will be discussed in the subsequent section of this chapter.
5.3 Image Compositing.

Figure 5.6: An image shot with a manual white balance setting of 2,800K. Note that in comparison to figure 5.4, that this appears to be the inverse. i.e., a cool colour appears blue. This is because the colour temperature given by the camera is a ‘correction’ of what would be applied under that light. i.e., a blue-shifted 2,800K image would appear white under orange candlelight.

Figure 5.7: The Colour-Channel Histogram for the scene at 2,800K.
5.3 Image Compositing.

Figure 5.8: An image shot with a manual white balance setting of 5,200K. Note this white balance is close to the ‘midday sun’ setting in figure 5.4, and is likely what a professional photographer would choose under the lighting conditions on the particular day and time. This image would normally be considered ‘optimal’, and is the image which compositing will most need to improve on for the technique to have any benefit.

Figure 5.9: The Histogram for the scene at 5,200K.
5.3 Image Compositing.

Figure 5.10: An image shot with a manual white balance setting of 10,000K.

Figure 5.11: The Histogram for the scene at 10,000K.
5.3 Image Compositing.

Figure 5.12: The colour composite image, taken as the per-pixel mean of the above three images. Note the colour difference in the ‘leafy’ regions between the two plants, especially compared with image 5.8. This composited image best matches the colour differences between the two actual plants.

Figure 5.13: The Histogram for the colour composite image. Whilst this histogram is similar to the 5,200K shown in figure 5.9, the three colour bands are more closely grouped in the 60-255 pixel range, signifying that the composite method has achieved a more even energy distribution across all three channels in comparison to the separate images.
5.3 Image Compositing.

5.3.2 High Dynamic Range Imaging (HDRI).

From (Elder 2004), the Dynamic Range of an image is the ratio between the largest and smallest possible values of the lighting that the sensor can detect. Unfortunately, while the human eye can typically detect a large dynamic range, it is difficult to reproduce this dynamic range electronically.

In terms of camera stops (a measure of differences in Exposure value [EV], with one stop equalling a doubling of light), the eye has a dynamic range of 10 to 14, while an LCD monitor can typically display only 9.5 stops. An 8-bit image typically records approximately 6 stops.

5 Objects can be seen under starlight as well as in bright daylight, even though at night objects receive $10^{-9}$ of the illumination received during the daylight; in power factor terms, this is a dynamic range of 90dB. (Elder 2004)

6 A good LCD monitor typically has a dynamic range of $10^{-3}$ (or 30dB) while some of the latest specialist CMOS image sensors have a measured range of 11,000:1 (40 dB) (Nikon 2014).

Figure 5.14: A side by side comparison of the histograms shows the effects of varying the white balance; in this particular image, the green channel stays relatively constant, while the blue and red balances shift. Note the very tight correlation in all three channels in the composite image.
In practical terms, this means that if a photograph is taken where the scene experiences a large difference in lighting between the lightest and darkest regions, often the camera will struggle to preserve the full data gamut. A classic example is an image taken indoors, with a window visible. While to the human eye both the data in the shadows and what is visible outside the window can be recognised, if a photo was taken of the scene one of two things would happen. Either the image would be exposed so that the shadow detail was visible, which would result in the window appearing as a single white block, or otherwise the detail outside the window would be recorded with everything within the room appearing as pure black. In this example, the first photo would be taken with a high exposure value (EV) number, while the second with a low (typically negative) EV. This effect can also be seen in image 5.12. The detail in the shadow regions has been lost as the sensor has recorded these regions as a pure black. At the same time, the tips of the flowers is approaching 8-bit saturation: adjusting the EV to show more detail in the shadow regions would result in the flowers appearing as pure white.

By taking multiple images over a range of EV, the complete lighting gamut can be recorded, with data visible in both the lightest and darkest regions of the scene. High Dynamic Range Imaging (HDRI) is a process of merging these multiple images together preserving the full gamut in one image. In a HDR image data is usually stored as a 32-bit floating point number per colour channel. Effectively, a HDRI records the total amount of light that has struck that region of the sensor, rather than simply a quantized 8-bit colour value.

In a typical HDR image, multiple photos (typically 5-9 total) are shot across an EV range and then composited into a single HDR file. EV ranges are typically +/-3EV for outdoor scenes, and may be as high as +/-13EV for extreme lighting differences (such as a dark room with a single window).

However, as most monitors are currently unable to display 32-bit floating point images, the results are usually tone-mapped back into 8-bit colour space. While this again results in a loss of data, careful tone-mapping can preserve the absolute
differences between colour and contrast intensities in the image. The figures below shows the entire concept of taking multiple exposures in order to increase the dynamic range, while the tone-mapped HDR image shows the result of careful tone-mapping to preserve the colour intensities.

It should be noted that in machine vision, there is no requirement to tone-map; if the processor is capable of working on the 32-bit floating point values then this would be preferable, however as this results in an extremely large dataset (roughly 150mb-1Gb per image) it is likely that the data will need to be tone-mapped to a smaller bit-depth for a phone processor to handle.

There are numerous dedicated software packages available for producing both a 32-bit floating point HDRI\(^7\), as well as ‘artistic’ packages\(^8\) capable of producing a tone-mapped image, while most SLRs and some compact cameras and mobile devices have automatic exposure bracketing (AEB) modes which can be used to capture the individual images in a single burst-fire. Together these tools largely allow for automating the HDR process. A number of high-end smart-phones and SLR cameras also include inbuilt HDR and tone-mapping functions for artistic purposes. Due to the common availability of both hardware and software, the derivation of the HDR algorithms will not be considered here.

---

\(^7\)Both Adobe Photoshop and Matlab have inbuilt HDR functions.

\(^8\)Such as HDRsoft’s “Photomatix” (hdrsoft.com 2014).
5.3 Image Compositing.

Figure 5.15: An image shot with 0EV. This image can be considered the ‘normal’ exposure setting which the camera would choose in automatic mode. Image shown is of *Nepenthes* leaf with stress spots due to heat/light intensity damage.

Figure 5.16: An image shot with -3EV. This image captures the data held within the brighter parts of the leaf, which would otherwise be over-exposed in a single-exposure image.
Figure 5.17: An image shot with +3EV. This image captures the data held within the darker parts of the leaf, which would otherwise be under-exposed in a single-exposure image.

Figure 5.18: The final composite HDR image after tone-mapping back into an 8-bit RGB colour space. By comparison with figure 5.15, the stress spots are much more pronounced, and therefore more likely to be detected in image segmentation stages, to be discussed in subsequent chapters.
5.4 Discussion.

This chapter discussed the effects of noise in an image, and developed methods to address this. These methods also served to increase the data available from low-end embedded cameras and aided in recovering some of the data from ‘lossy’ file formats such as JPEG. From the available literature, it seems that the process of white balance compositing presented here is an entirely innovative approach. Whilst over-sampling and HDRI are both established photographic techniques, their usage in machine vision applications is also considered innovative.

5.5 Conclusions.

Image augmentation techniques can successfully be used to improve on the data recorded by the camera, compared to just relying on a single captured image. Over-sampling techniques and their derivatives can effectively be used to improve on the ‘quality’ of the camera itself (e.g., the camera in the Galaxy Tab 3 was found to be inherently noisy). However, whilst these techniques have merit in the current disease analysis application, the disease examples supplied by the agronomists were only captured as single images. The rest of this dissertation focuses on the use of these ‘simple’ images, with the augmentation methods discussed in this chapter again being addressed on actual photos in the evaluation chapter 11. Further uses of HDR and RAW techniques for disease recognition are also discussed in appendix F.
Chapter 6

Leaf Extraction from Images.

6.1 Chapter Overview.

In any given image, there is often a large portion of space that effectively counts as ‘wasted pixels’. Subsequent steps in image processing are often memory-intensive, and as such regions of the image which do not contribute to the task at hand need to be removed. This is the task of image segmentation. One of the first tasks to be undertaken in any vision application is to extract, or segment objects from their backgrounds. Whilst this is a seemingly innocuous task, much research has been spent in previous decades to identify the most efficient methods (Davies 2012), (Kumar, Torr & Zisserman 2010).

Figure 6.1 shows a typical example of image segmentation. It can be seen that as much as 80% of the original image has been eliminated, leaving a single leaf for subsequent segmentation and feature extraction stages.
6.2 Stereo Motion Capture.

Traditionally (Davies 2012), stereo vision relies on two separate cameras mounted a fixed distance from each other. By using the stereo or binocular outputs from both cameras it is possible to build up a 3-dimensional composite image from which depth perception can be extracted.

It was anticipated that as the diseased leaf in question would likely be the object closest to the camera lens, the use of depth perception as a means of segmenting out the background and other leaves would be possible. However, as only a single camera was to be used research into ‘stereo from motion’, i.e., taking two or more images using a moving camera to build up the depth channel was investigated\(^1\).

When an imaging device is hand-held during long exposure times, the captured

\(^1\)Davies (2012) briefly mentions this approach, but only considers a ‘dolly’ camera, i.e. one that moves in and out along the camera lens-axis.
image is often blurry. This is attributed to the muscles in the hands being unable to accurately hold the camera steady for longer than a few tenths of a second. It was proposed to deliberately use this phenomenon to introduce relative motion into the image capturing stage by taking multiple separate short-exposure images at approximately 1/2 second apart.

Figure 6.2 shows\textsuperscript{2} the basic premise of stereo analysis: by taking multiple (at least two) images of the same scene, it is possible to mathematically establish the camera positions from the data-points in the scene, and from there ‘reverse engineer’ the scene back to a single 3-dimensional scene. Figure 6.3 shows two images captured of the original scene. Mathematically these images are reshaped until all points corresponding between the left and right image lay on a horizontal line. This is seen in figure 6.4. Figure 6.5 shows the depth map which is reconstructed from the scene: here the brightness of the greyscale image indicates the depth of objects in the scene: by choosing a single greyscale level (i.e. closest to the center of the image, corresponding to the ‘point of interest’) it is possible to use this depth map to mask out a single leaf from the surrounding field. See Appendix C for a comprehensive derivation of the mathematical process associated with this technique.

\footnote{The CGI images shown in figures 6.2 through 6.5, and appendix C were created by the author using “Autodesk 3DS Max 2010” software. The depth map in figure 6.5 was created by using the software’s own internal depth-map capability and is intended to represent the stereo concepts, rather than a proof of the machine vision extraction methods contained in this chapter.}
Figure 6.2: Shows the ‘complete picture’ of stereo reconstruction. **A** at the left of the image shows the actual field with crops (in 3-dimensional space). **B** and **C** show two images captured of the same plant (respectively, **B** and **C** are the left and right images). In **D**, the two images have been calibrated so that all points visible in the two images are aligned horizontally between left and right. After this calibration stage, it is down to trigonometric relationships to compute the relative distances of all points in the image: the tip of the plant is closest to the camera due to the camera (**B**, **C**) angles, thus in the depth map **E**, the tip and leftmost leaf have the brightest greyscale level, denoting the relative distance to the cameras. (All CGI images generated by the author.)

Figure 6.3: Left and right images of the same wheat sample. Here the positions of the cameras are currently unknown, however by taking numerous points such as leaf tips that correspond between images, it is possible to solve for the unknown degrees of freedom and thus compute the camera positions from the data contained within the scene. Note that in actual use the infected leaf would take up more of the image frame than in this example.
6.2 Stereo Motion Capture.

Figure 6.4: Once the camera positions are found between images, it is possible to reshape (twist, skew) the two images themselves so that all corresponding points lay on a horizontal plane between images. This makes subsequent steps in searching for matching pixels across images easier, and thus aides in the construction of the depth map.

Figure 6.5: Once the images have been reshaped and all corresponding data-points between the two frames located it is possible to use trigonometric relationships to ‘reconstruct’ the depths (in 3D space) of all points in the image. This reconstructed image can exist either as a point cloud in 3D space within the software, or as here be used to generate a Z-depth map of the scene. Note that the upper tips and left leaf are brightest in this image. Comparing this with the above figures it is evident that these points are closest to the camera. By choosing only the whitest regions, this Z-depth image could then be used to extract a single leaf from the original images.
In order to provide sample images for stereo extraction experiments, an agricultural site was visited, and sample photos of the growing wheat were taken. Two experiments on these images were conducted: initially, camera shake extraction was trialled to prove the concept of taking two photos at 1/2 second intervals and extracting depth from this stereo pair. A second pair of photos were also taken with the camera having moved approximately 75mm horizontally between exposures, in order to enhance the parallax effect between images. The results of this second set of images will be covered in a subsequent section of this chapter. Finally, samples of wheat were grown at home to provide continual access to live plants for imaging experiments. Experiments on these samples will also be covered below. It is worth noting however that none of these samples were infected with disease: the experiments in this chapter focussed instead purely on methods for single leaf extraction.

### 6.2.1 Camera-Shake Extraction.

The first process Stereo depth extraction is image rectification, followed by construction of the disparity depth map, which compares points directly between images.

#### Image Rectification.

The two wheat photos are shown below in figures 6.6 and 6.7. Both images were captured at the same time, using the camera set to burst-fire mode with a half-second interval between exposures. All relative motion between images is due to the movement of hands and wrists over the 1/2 second interval.
6.2 Stereo Motion Capture.

Figure 6.6: The first burst-fire image. The wheat crop in these photos is approximately 4 weeks old. Image captured with a Kodak EasyShare Z740 5MP camera.

Figure 6.7: The second image, shot at a 1/2 second interval after the first, shown above.

The initial step was to convert both images to grey-scale: Using the colour channels to compute depth is unnecessary and working on a single colour channel image is less computationally intensive. Figure 6.8 shows the grey-scale images.
6.2 Stereo Motion Capture.

side by side, while figure 6.9 shows both overlaid, with the left image given a red tint while the right image is coloured cyan.

Figure 6.8: Both left and right images converted to grey-scale, and placed side by side.

Figure 6.9: Left and right images overlaid. The left image is given a red tint, while the right image is coloured cyan. This colour is for channel identification purposes only and has no bearing on the stereo algorithm. Note the higher parallax (distance between channels) in the leaves compared to the ground, indicating they are closer to the camera in this stereo composite.
6.2 Stereo Motion Capture.

The greater difference in red/cyan around the leaves in figure 6.9 indicates a larger parallax, since these leaves are closer to the camera than the background. In order to merge these images and extract the depth, it is first necessary to rectify the images. Rectification involves locating matching salient points across both left and right images, and using these first to compute the epipolar points, and from this computing the fundamental matrix. The fundamental matrix is then used to transform both left and right images so that every corresponding point is aligned horizontally. This process is shown below, while Appendix C explains the construction of the fundamental matrix in detail.

Figure 6.10 shows 500 salient points found in each image, by using corner detection. A large number of points needs to be identified between each image. Figure 6.11 shows the result of matching correspondences between both images: a $9 \times 9$ pixel block around each point is sampled, before cross-correlating this $9 \times 9$ block with the points in the opposite image. It is notable that significantly less than 500 points have been correlated: it is necessary to detect a large number of salient corners to ensure there are enough correlations.

Figure 6.10: 500 points are located using corner detection in both left and right images.

From the disparities between matched points, it is possible to compute the fundamental transformation matrix for these images\(^3\). The result of applying the fundamental matrix to the images is shown in figures 6.12 and 6.13. Figure 6.12

\(^3\)Refer Appendix C.
6.2 Stereo Motion Capture.

Figure 6.11: The matching points located between both image 1 and 2. A short vector (yellow) shows each of the disparities between points. Already at this stage it is evident that the vectors all seemingly point to an imaginary location in space.

shows the overlaid rectification results: each of the matching points now coincides between images. Figure 6.13 shows both images side by side, with each of the epipolar lines drawn in. Note that due to rectification, the epipolar lines are all now parallel, and also line up horizontally between left and right images. This is important for the subsequent depth extraction process: with all corresponding points laying in parallel planes, depth can be extracted using trigonometric relationships.
Figure 6.12: The result of computing the fundamental matrix, and applying it to both images. The images have been transformed so that the matched (inlier) points correspond between two images.

Figure 6.13: Side by side images. Due to the rectification (stretch and rotation) the epipolar lines in both images are now parallel, and all points correspond horizontally between images.
Construction of the Depth Disparity Map.

By rectifying the images in the preceding section, all corresponding pixels between both left and right images lay in a horizontal plane, allowing depth to be computed by trigonometric relationships. The importance of rectification is best shown diagrammatically: Figure 6.14 shows that by computing the fundamental matrix and rectifying the image, that the actual real-world distance between the two cameras, and the camera image planes can be computed. By rectifying the two images, it allows all data-points in each image to be found on a single plane. Depth of each pixel in the image can then be found using simple trigonometry.

![Diagram of stereo motion capture](image)

Figure 6.14: Once the fundamental matrix has been applied, both images are rectified, i.e., the distance between the cameras has been computed, as has the focal plane depth. By stretching and rotating the images, all points now exist on the common plane. By locating matching points between images, the real-world depth from each pixel can be computed.

The black regions in figure 6.13 contains no usable data, while the red region in the lower left region of that figure shows pixels which aren’t present in the edges of image 2. Since only pixels common to both images are of importance, these regions can be cropped, as shown in figure 6.15.
As each corresponding point between the two images is now aligned horizontally, a $7 \times 7$ pixel window is used to search along the horizontal axis. When a matching $7 \times 7$ pixel block is located in the opposite image, the relative depth is computed. The complete depth map is shown in figure 6.16.
Figure 6.16: The complete depth map. The blue regions denote a greater distance from the camera, while red regions are closer. Note the dark red region in the lower half of the image. This is due to the foreground leaves in the lower half of the image being closer to the camera than the two main upright leaves. The green colour of the two upright leaves indicates correct depth extraction, although this image is very noisy.

Figure 6.16 shows the final depth extraction; the green regions to the left of center correspond to the two crossed leaves visible in figure 6.15. Note that while the process shows the beginning of a usable method, that the output is very noisy. This occurs when there are no strong image features within the $7 \times 7$ pixel window, leading to uncertainty as to the absolute location (and hence depth) of each pixel in the $7 \times 7$ window. (MathWorks, Inc. The 2010).
6.2 Stereo Motion Capture.

6.2.2 Camera Movement Extraction.

The results in the previous section were encouraging, however given the results from the disparity map, it is likely that using camera shake alone doesn’t provide enough movement to construct an accurate depth map.

The process was repeated, using two photos taken with an estimated 75mm of movement between them, shown in figures 6.17. As the depth extraction process was identical to that explained above only the results will be shown. Figure 6.18 shows the left and right grey-scale images, while figure 6.19 shows the overlaid images after correlation and cropping. By comparison to the images in the preceding section, the parallax is much greater with the larger movement.

Figure 6.17: Images taken with approximately 75mm camera movement between exposures

Figure 6.18: Left and Right Stereo Images - 75mm Movement
Figure 6.19: Colour overlay of the correlated and cropped images.

Figure 6.20 shows the final depth map. By comparison with the results above the leaves are more prominently extracted, however on close inspection there appears to be a 'double exposure' in the two main vertical leaves (i.e., the red and cyan leaves in figure 6.19 have both been extracted.) This possibly indicates that 75mm movement was excessive, and the $7 \times 7$ block-matching was unable to detect that these leaves were the same item.
6.2 Stereo Motion Capture.

Figure 6.20: The resultant depth map. Red colours show points closest to the camera, while blues shows more distant points. It is evident that the increased motion between photographs has given increased depth recognition, however there is a degree of ‘double exposure’ present, as the block matching was unable to recognise the leaves as the same same item.
6.2 Stereo Motion Capture.

6.2.3 Extraction - Grown Samples.

The above experiments were conducted on wheat which was still in the ‘tillering’ stage, i.e., the basal leaves had emerged, but the main stem had not elongated. As such, the leaves were close to the ground and this proximity gave a noisy output which would need subsequent filtering. In order to find a general solution which could work when the plants had matured, wheat samples were grown at home. The subsequent photographs were taken when the wheat was 20 weeks old, and the ear (grain) had just started to develop. As the depth procedure is identical to the methods explained above, it is sufficient to show just the input images in figure 6.21, and the results of depth mapping in figure 6.22. Camera motion between the two photographs was approximately 50mm.

Figure 6.21: Left and right images, with approximately 50mm motion between captures. The wheat in these images is 20 weeks old.
6.2 Stereo Motion Capture.

Figure 6.22: Depth extraction from the actual wheat sample. Note the part of the hand holding onto the leaf has also been detected at the same depth as the leaf.

By comparison with the basal leaf extraction samples, the background is (as expected) far less noticeable. However, there is still a ‘double exposure’ in the top leaf, and critically, the part of the hand visible in the top left corner of the photos (used to pull the leaf out straight) as also been recognised at the same depth as the leaf. This indicates the current algorithm is very sensitive to stray objects at the same depth as the leaf and would need to be improved for this depth extraction method to be viable.

6.2.4 Stereo Extraction - Conclusions.

With improvement of the algorithms, stereo extraction may prove a viable solution. However as the initial salient point detection\(^4\) uses corner finding methods, and there are more corners present in the background than in the leaves themselves it is possible that the fundamental matrix will always compute for the background first and the leaves second, resulting in ill-defined results. The lack of fine detail within the leaves may also be responsible for the ill-defined

---

\(^4\)See figure 6.10.
results of the $7 \times 7$ pixel block search. The Matlab Computer Vision Toolbox documentation\textsuperscript{5} indicates that “when no strong image features appear inside of the 7-by-7-pixel windows being compared. Then the matching process is subject to noise since each pixel chooses its disparity independently of all the other pixels.” Suggestions in that reference such as using sub-pixel estimation rather than block searching failed to improve the results over the depth maps shown in this chapter. In conclusion, it was decided to investigate other methods of depth extraction, as explained below.

### 6.3 Manual Depth Extraction.

In the preceding section, a fully automated leaf extraction method was trialled. This section discusses the use of a semi-automated method, using an 18\% gray reference card placed behind the leaf before taking the photograph\textsuperscript{6}. With the image appropriately cropped so that nothing is visible outside the gray reference, it is comparatively much simpler for the software to focus simply on the target leaf. Figure 6.23 shows the overall process on the grown samples, while figure 6.24 shows the target leaf after manual cropping so that just the gray card region is visible. Figure 6.25 shows the final result of the software extracting the gray region from around the leaf.

\textsuperscript{5}(MathWorks, Inc. The 2010).

\textsuperscript{6}18\% gray white balance reference cards were briefly discussed section 5.3.1 on page 35.
6.3 Manual Depth Extraction.

Figure 6.23: Manual Extraction: Use of 18% gray reference card. By placing the card behind the object of interest, the software can extract just a single leaf.
Figure 6.24: A single leaf with the reference card behind it. The software can easily recognise the large gray regions and crop them out of the image. (Note the leaf has been uncurled in this photo, compared to the one above.)

Figure 6.25: The single leaf, with all background items removed. The use of the 18% gray reference card also allowed for quick white balance correction to remove the blue tint from the image just prior to deleting the background, as discussed in section 5.3.1.
6.4 Leaf Extraction - Conclusions.

Manual leaf extraction with a reference card allowed for a quick, accurate, and semi-automated extraction process, with crisp edges to the leaf boundaries and a successful extraction. However, as this method still requires the use of ‘props’ and thus some manual involvement, it is desirable to continue investigations into a completely automated process which can identify and correctly segment one leaf in-situ.

Manual extraction is a perfectly suitable method for isolating one leaf for the subsequent diagnosis phases, and is the method used in the following chapters. However it is considered that a fully automated method will be a topic for further research and will be discussed again in chapter 13.
Chapter 7

Disease Spot Extraction from Leaf Regions.

7.1 Disease Sub-segmentation.

Once a single leaf has been extracted as the area of interest, it is necessary to begin extraction of the diseased spots from the leaf for further analysis. Investigating available techniques highlighted two approaches which will be analysed in subsequent sections: Colour region masking using histogram analysis, and Statistical Pattern Recognition using K-means clustering. Colour Region masking was explained in (Rosenfeld & de la Torre 1983). The K-means approach was described in (Davies 2012) and extended by (Tan et al. 2006) and the Matlab help files (MathWorks, Inc. The 2010).

As different image capture devices have different grades and resolutions of sensors\(^1\), a comparison of the effects of varying the resolution of the captured image will also be investigated in the relevant sections of this chapter to discern if an image from a lower quality camera can still be used with the same sub-segmentation techniques as with a higher resolution device.

\(^1\)Refer chapter 5.
7.1 Disease Sub-segmentation.

7.1.1 Colour Region Masking.

RGB Colour Space

The image as captured by the camera is represented in a colour space consisting of Red, Green and Blue (RGB) channels, each consisting of 8-bits. Any pixel in the image is thus represented as a combination of these 3 channels: a light-coloured pixel has high-values in all 3 channels, while a dark coloured pixel has low values. A light green region of the leaf will have a large value in the green channel with some red and blue values included, while a darker region of the same leaf such as a region in shadow will have darker values. Figure 7.1 shows a typical rust-infected leaf, with the separate RGB channels that are combined to produce the complete image.

![Figure 7.1: A colour image of a leaf rust infection on the left, and the individual Red, Green and Blue channels of the leaf. Note that there is very little information contained in the blue channel, and also that most of the diseased region shows up strongest in the red channel. (Leaf image supplied by Dr Neate.)](image)

HSV Colour Space.

Although the camera initially captures the image in RGB colour space, it is often advantageous\(^2\) to convert the image into another colour space. One such

\(^2\)(Davies 2012)
7.1 Disease Sub-segmentation.

colour space which is useful for machine vision applications is Hue-Saturation-Value (HSV). This colour space represents the colour of a pixel in an alternate way; a comparison between dark and light regions of the same leaf will in general have similar Hue values, and disparate Value (or Brightness) components. For example, a leaf which is in partial shade should have a consistent shade in the hue channel, while the other channels contain the variations. Thus, it is the hue channel alone that contains all colour information. As such, by converting the original JPEG image from RGB to an alternate colour space it is possible to simply ‘mask out’ any green regions of the leaf, in the hue channel leaving only the diseased regions behind. Figure 7.2 shows the hue, saturation and value channels of the same leaf shown in figure 7.1. It is evident that the hue channel contains the most useful information for spot extraction, as the darker regions represent the infected parts of the leaf: by setting a threshold on this channel, any parts of the leaf which are above the threshold can be masked out, leaving just the spots behind.

![Image](image.png)

Figure 7.2: From left to right: Original image, Hue, Saturation, Value channels. Broadly speaking, the hue channel contains all colour information. Saturation controls the white intensity, while the value channel controls the light-to-dark. For the given application, just the Hue channel can be used to extract the individual spots.
7.1 Disease Sub-segmentation.

Histogram Masking.

After extracting the single hue channel, the question remains as how to best mask out the regions. One of the simplest methods is to use a histogram. The basic premise of the histogram is that it counts up all the individual pixel values in the leaf image, and displays the total number in ‘buckets’, i.e., it counts up the total number of pixels in the image which share the same numeric value. Figure 7.3 shows the Histogram of the Hue channel from figure 7.2. From the histogram it can be seen that there is a local minima around the 0.15 mark. This means that most of the pixel values in the image below this local minima likely represent the red diseased regions, while those above this number are the green healthy regions.

![Histogram of the hue channel. Visual inspection of the hue channel from figure 7.2 suggests the leaf can be simply considered as two regions: ‘light’ and ‘dark’. Inspecting the Histogram reveals that there is a suitable divide into these two regions at the 0.15 value.](image-url)
By simply setting all values above 0.15 to zero, the Hue channel can be masked out to leave just the diseased regions. Applying this mask to the original RGB image gives the results seen in figure 7.4: all the healthy regions of the leaf have thus been discarded.

Figure 7.4: Results of masking out the histogram regions on the hue channel, and reapplying this mask to the original RGB image. The healthy regions have been discarded, leaving just the infection regions for further analysis and classification.

7.1.2 Effects of Resolution.

It was mentioned in chapter 5 that the cameras used in the sampling had a widely different resolution (Mega-pixel) and sensor quality range. As well as this, the leaf image samples obtained from the agronomists were generally of low quality. In order to test the effects of using lower resolution imaging devices with the
above described sub-segmentation methods, it was necessary to repeat the above comparisons, using images which were of a deliberately lower resolution than those used in section 7.1. This was accomplished by taking a higher resolution image and artificially blurring it with a Gaussian filter a set number of times before applying sub-segmentation. The results of this are seen in figures 7.5 and 7.6, for a $5 \times 5$ Gaussian filter convolved with the input image respectively, 50 and 100 times\textsuperscript{3}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{7.5.png}
\caption{Results of blurring the leaf sample 50 times with a Gaussian filter. 33 unique spots were found.}
\end{figure}

\textsuperscript{3}Filter Convolution is covered in detail in chapter 8.
7.1 Disease Sub-segmentation.

Figure 7.6: Results of blurring the leaf sample 100 times with a Gaussian filter. 29 unique spots were found.

From the results, it is clear that as the level of blur increases, the definition between spots decreases: spots which were not connected in the original segmentation in figure 7.4 become conjoined, so that the total number of separate spots decreases. In the original image, the software identified 38 unconnected spots, while figure 7.5 identified only 33. Figure 7.6 identified 29 spots. From this the conclusion is that image resolution only has a minor effect on the number of spots identified, however the spots themselves are inevitably blurred and thus contain less texture data: the relevance of texture data to frequency analysis will be discussed in chapter 8.
7.1 Disease Sub-segmentation.

7.1.3 K-means Clustering.

Whilst the Histogram approach in the preceding section gave good results, there is still scope for improvement: although the histogram value of 0.15 seemed to work fairly consistently across most leaf samples, it was still derived manually. For a fully automated solution, an unsupervised approach was also considered.

K-means clustering\(^4\) is one of the simplest unsupervised, non-iterative grouping algorithms. Conceptually, it can be viewed as follows: random points on the image are assigned as centroids, and then each pixel in the image is grouped according to which centroid it most closely matches. The centroid is then moved to the mid-point of this first step of clustering, before the process is repeated until there is no further refinement of the clustering steps.

The outcome thus involves partitioning the image into a prescribed set of clusters, by grouping individual pixels (and neighbouring regions) in the image according to the cluster that they best match. The number of clusters is usually determined \textit{a-priori}: in simplest terms, this could be done with two clusters: one for the diseased sections of the leaf, and one for the non-diseased parts.

Despite the simplistic approach, K-means clustering suffers from two major drawbacks. The first is that the number of clusters needs to be determined before the simulation is run: normally this is handled by deliberately assigning more clusters than necessary and accept that some clusters will be empty. In practice it was found that more than two clusters was needed, as some diseases (rusts) have both light and dark regions of the same spot. Other diseases (tan spot) visually have both a light tan region representing the sick portions of the leaf, and dark brown regions representing necrotic tissue. Variations in lighting also contributed to inaccurate segmentation with small regions; a total number of no less than four clusters was required for best grouping.

The larger issue however with K-means clustering is that the randomisation step in the first instance necessarily increases the likelihood of two centroids being

\(^4\)From (Davies 2012), and Matlab’s inbuilt functions.
assigned to the same region, e.g., in the case of a leaf with few spots it is very likely that the healthy region may be assigned more than one cluster. The result is often an ‘empty’ cluster at the end of the segmentation as moderately similar (i.e. light green and light tan) regions of the leaf are grouped together. It is this sensitivity to initial conditions\(^5\) which causes the most problems with this method. This problem was solved in Matlab by performing a preliminary randomized clustering phase, using 10% of the original dataset, and by creating a new cluster consisting of the one point furthest from the respective centroid, if any of the initial clusters was empty. Both of these functions are inbuilt extensions to the basic Matlab K-means algorithm.

**CIELAB Colour Space**

For K-means to work effectively on the leaf dataset, it was necessary to again transform the original RGB image into another colour space. However, initial experiments with HSV colour space were disappointing, with poor accuracy in detecting spots. Further research indicated that using CIELAB colour space would yield improved performance. CIELAB\(^6\) Colour space involves the use of three channels: the Luminance Channel, and two opponent-axis channels, with the \(a\) channel handling the green-red band, and the \(b\) channel the blue-yellow.

Adobe Systems Incorporated (2000) indicates that the LAB colour space was designed to mimic the retinal colour stimuli in the human vision system, as the non-linear response of the eye to changing colours is reflected in the colour space, i.e., CIELAB uses non-linear gradients between colours in the \(a\) and \(b\) channels to better match human colour perception.

\(^5\)(Tan et al. 2006)

\(^6\)Alternatively CIE-L\(^*\)a\(^*\)b. from the Commission Internationale de l’clairage
Figure 7.7: Graphical representation of the CIELAB colour space. This colour space closely matches the way that the human eye perceives colour, since a colour cannot be both red and green ($a$-axis), or blue and yellow ($b$-axis) at the same time. The vertical axis contains the illumination data, while any colour can be represented as a combination of the $a$-$b$ axis. (CGI by author.)

Figure 7.8 shows the effects of converting the leaf into LAB colour space. Note that Luminance channel now contains all variations in lighting across the image, while the $a$ channel shows the strongest variation between green and red between the diseased and healthy regions. The $b$ channel also shows a strong response in the most yellow regions of the leaf. As discussed above, there is little blue in the leaf sample, hence the $b$ channel doesn’t show as much variation as the $a$ channel.
7.1 Disease Sub-segmentation.

Figure 7.8: Left to right: Luminance channel, $a$ channel, $b$ channel. Note that all illumination variation is contained in the $L$ channel, while the $a$ channel shows the strongest variation between green and red. Most of the data in the $b$ channel is in the yellow region.

Applying K-means clustering to both $a,b$ channels with four clusters yields the results seen in figure 7.9. It is evident that the diseased spots have been clustered equally into the white region, while the yellow and green regions have respectively each received a cluster. The final cluster is 'wasted' on the black background. Note that particularly in clusters 2 and 3 that there is a high degree of sensitivity partitioning between the healthy green leaf, and the slightly yellow regions which denote regions where the plant’s immune system is fighting the infection.
7.2 Discussion: Comparison of Methods.

Both colour region masking and k-means clustering perform similar tasks, however the approaches are quite different. Comparing the results of the two methods shows that region masking is quick and efficient, but the reliance on fine-tuning the threshold means that it is less robust. Conversely, K-means clustering provides a general solution which works across a large range of leaves, but the conversion to CIELAB colour space and the k-means algorithm itself are
7.2 Discussion: Comparison of Methods.

computationally intensive. Figures 7.10 and 7.11 show a comparison between the two methods across different leaf samples, using two threshold levels (0.13 and 0.15 respectively) for region masking, and the same k-means approach for both leaf samples.

Figure 7.10: Left to right: The original leaf image. Colour region masking with a threshold of 0.13. Masking with a threshold of 0.15. K-means clustering.

Figure 7.11: Left to right: The original leaf image. Colour region masking with a threshold of 0.13. Masking with a threshold of 0.15. K-means clustering. Note that the dermal tissue where the spores have burst through the leaf surface has also been removed in the K-means approach. This is of less concern since the final spot extraction will include portions of the original leaf image. (Leaf image supplied by Dr Neate.)

In general with the region masking, a slightly conservative value for the threshold results in a tighter cropping around each of the spots, however care must be taken
7.3 Region Extraction.

to ensure that actual regions of interest in the spots themselves aren’t cropped out. Using the K-means approach this isn’t an issue, as the more robust approach works equally well for both samples.

Unfortunately however, despite the advantages of using K-means, it is necessary to consider the time factor: The entire process of converting to HSV and masking took approximately 0.14 seconds in Matlab, while converting to CIELAB and performing K-means took 2.2 seconds. Despite the increased time using K-means, it was decided to use this approach for the later stages at least during the testing phase of this project, as the increased robustness of the method outweighed the extra processing time.

7.3 Region Extraction.

As a final step in preparing the disease spots for later chapters, it is necessary to discuss how the spots are extracted. As classification using Neural Networks\textsuperscript{7} often requires some healthy region of the leaf added in for training purposes, it was decided to use the boundaries of the extracted spots simply as masking regions, and extract spots from the original RGB images. Figure 7.12 shows the results of this: The regions from figure 7.11 were used to generate the red bounding boxes. Each of the RGB regions enclosed in a red boundary was individually extracted as a separate object, and then rescaled into a $32 \times 32$ pixel image, ready for classification. Note in this final image, the connected spots (i.e. more than one spot per extracted region) are clearly shown. Whilst one of the purposes of sub-segmentation was to extract individual spots, in many diseases the infected regions become conjoined: using both these conjoined and separate spots in the classification stages will allow the software to ‘learn’ that this is the same disease.

\textsuperscript{7}See chapter 10
Figure 7.12: On the left, each of the regions surrounded in red is extracted as a separate object, while on the right each spot is rescaled into a $32 \times 32$ pixel image ready for the subsequent classification steps, discussed in the next chapter.
7.4 Conclusions.

This chapter has shown alternative approaches to extracting the diseased regions from the crop leaves, as well as shown the effects of image resolution on the extraction processes. Whilst the K-means approach was shown to take longer to process in software than region masking, it was decided to use this approach in subsequent stages, as the increased robustness of the method outweighed the extra processing time. Finally, the method of extracting each spot as a $32 \times 32$ pixel image was discussed. The separate images will be classified in the subsequent chapters.
Chapter 8

The Gabor Filter.

8.1 Chapter Overview.

In recent years, the concept of the Gabor filter has gained a lot of traction as a strong classifier of both texture representation and discrimination. It has been used to very good effect in machine recognition of faces, fingerprints, handwriting and text\(^1\), as well as iris detection for identity recognition. Daugman (1985) discovered that the simple cells in the visual cortex of mammalian brains can be modelled by Gabor functions, while Jones & Palmer (1987) showed that the real part of the complex Gabor function is a good fit to the receptive field weight functions found in simple cells in a cat’s striate cortex. It is this combination of biological, as well as mathematical motivation that makes the Gabor filter of interest in machine vision.

The strength of the filter comes from its structure; from Daugman (2004) in iris recognition, the multi-scale quadrature (complex number) structure of the Gabor wavelet makes it extremely sensitive to changes in frequency (dimensionality) and rotation (orientation) of the base image that the filter is applied to. As such, using a filter wavelet with a tunable kernel allows it to perform multi-resolution

\(^1\)Optical Character Recognition, or (OCR)
8.2 The Gabor Filter Bank.

analysis of the input. Appendix D gives examples of this ability to selectively tune the filter to detect specific elements in the test images, as well as background information on the mathematical derivations behind the filter construction.

This chapter and companion appendix draws on the work of (Daugman 1985), (Daugman 2004), and (Chao 2010). This chapter also presents an alternative approach to the classic Gabor filters, as outlined by (Field 1987) and (Kovesi 2013) by constructing the filter in logarithmic space.

Traditional approaches for applying Gabor filters to the input images use Filter Convolution in the Spatial domain. This chapter also presents the novel approach of applying the basic signal analysis approach of filtering in the frequency (Fourier) domain, and presents a comparison of the improvements in processing time associated with this variation.

8.2 The Gabor Filter Bank.

The basic Gabor filter bank is shown in figure 8.1. This filter bank consists of 40 filter wavelets, with 8 variations in rotation (i.e., the columns in the figure) and 5 variations in frequency. By applying each of these 40 filters to the input image (i.e., the disease spots extracted in chapter 7), a complete filter response can be constructed which is similar for each spot of a particular disease, but disparate when compared with samples from an unrelated infection. The derivations of the individual Gabor wavelets which make up the bank are explained in Appendix D, while relating the specific filter responses to the particular diseases is covered in subsequent chapters.
In order to perform multi-resolution analysis, it is necessary to use a tunable kernel which can be varied in both frequency and orientation. Appendix D.1 explains that by using an appropriate bank of filters, that any features within an image can be uniquely extracted. The question remains then, as to what constitutes an ‘appropriate’ filter bank? In order to answer this question, it is necessary to consider the Gabor filters in frequency (Fourier) space. A detailed explanation of Fourier conversion of the spatial Gabor wavelet is found in Appendix D.2. A typical conversion of the 2D Spatial wavelet into a 2D frequency wavelet is shown in figure 8.2.
In order to construct a filter bank which covers the full range of both frequency and orientation, multiple filters need to be used. By superimposing each of the Fourier-transformed wavelets on top of each other, the complete filter bank can be optimised so that each wavelet detects a particular frequency and orientation, while combined the complete filter bank covers the full spectral range of the input image, allowing for every salient feature in an image to be extracted. This is explained in figure 8.3. The choice of the bandwidth in orientation is motivated by the simple cell orientation resolution which has been evaluated as around 20° to 40° of full bandwidth at half response\(^2\). This typically requires orientations of between 6-13 to cover the full 180° range on Fourier space, with a value of 8 orientations being the norm.

The bandwidth in frequency is usually chosen as between 0.6 and 3.2 octaves, and around 1.3 octaves is the mean (Daugman 1985). A value of \(\sqrt{2}\) was chosen as it gave good overlap with a total number of 5 filters in frequency. These two conditions are met in relationship 8.1.

\(^2\)(Fischer, Šroubek, Perrinet, Redondo & Cristóbal 2007)
8.2 The Gabor Filter Bank.

\[ \{ \varphi_{\text{discrete}}(f_u, \theta_v)(x, y) \} \]  

\[ f_u = \frac{f_{\text{max}}}{\sqrt{2^U - 1}} \bigg|_{u=0} \quad \theta_v = \frac{\pi}{V - 1} \bigg|_{v=0} \]  

(8.1)

Figure 8.3: Figure shows the effect of using multiple filters in both Scale/Frequency \((U = 5)\) and Orientation \((V = 8)\) to cover the full range of the frequency domain. Overlaps between individual filters should be large enough to permit even coverage of the spectrum, while at the same time being small enough to ensure they are as independent as possible. Adjusting the ellipticity with \(\theta, \eta\) permits tuning for broad or narrow-bands separately in frequency and orientation. The low-frequency region in the center of the rosette can be covered with a separate low-pass filter if necessary.

As such the filter bank can be constructed using the relationship 8.1 in the Fourier transform equations\(^3\). The resultant filter banks can be seen in figures 8.4 and 8.5, Values used are: \(f_{\text{max}} = 2, U = 5, V = 8, \theta, \eta = 1.2\). Figure 8.6 shows that the multi-resolution rosette is satisfied.

\(^3\) equations D.4 and D.7, see pages 224 and 232 respectively.
8.2 The Gabor Filter Bank.

Figure 8.4: Figure shows the complete Spatial filter bank, with 5 frequency and 8 orientation variations. Values used in the bank’s construction were: $f_{\text{max}} = 2$, $U = 5$, $V = 8$, $\theta, \eta = 1.2$

Figure 8.5: Figure shows the complete Fourier filter bank, with each discrete wavelet related to its counterpart in figure 8.4.
8.3 The Gabor Filter Usage.

8.3.1 Filter Convolution.

Traditionally\(^4\), use of the Gabor filters uses convolution as the method for applying the filter to the input images. Convolution is a useful concept that has many applications in engineering and image processing, and is often at the core of any smoothing or sharpening functions in signal processing. The definition given in Davies (2012) states that: given two piecewise-continuous functions \(f(x)\) and \(g(x)\), spatial convolution\(^5\) is described as the integral:

\[
 f(x) * g(x) = \int_{-\infty}^{\infty} f(u) g(x - u) \, du
\]

The action of this integral is normally described as a travelling filter function

---

\(^4\)See the literature review, chapter 3.
\(^5\)Note here that the * symbol denotes convolution rather than multiplication.
8.3 The Gabor Filter Usage.

\( g(x) \) overlapping the input signal function \( f(x) \) with a travelling step size of \( du \) and a travelling offset of \( u \). Graphically, this can be shown as in figure 8.7

![Graph showing principal of spatial convolution](image)

Figure 8.7: Figure shows the principal of spatial convolution. The black solid line represents the input \( f(x) \). The blue dotted line is the filter \( g(x) \). By moving the filter to the right over the image with step size \( du \), and plotting the combined area under the curve intersections at each interval \( u \), the convolved output can be derived, shown in red. Using a Gaussian function for \( g(x) \) results in a ‘smooth’ filter response: this application is typical for blurring images, and is used to smooth sharp inputs without degrading the overall shape or area of the signal.

When convolution is applied to digital images, the convolution equation changes in two ways: a) Since the image is 2-dimensional, a double integration must be used, and b) the integration must be performed over the discrete interval bounded by the size of the image and filter. The resultant 2-dimensional convolution integral is thus:

\[
F(x, y) = f(x, y) \ast g(x, y) = \sum_i \sum_j f(i, j) g(x - i, y - j)
\]

where \( f(x, y) \) denotes the input image, and \( g(x, y) \) is the spatial convolution filter. Coordinate system \( i, j \) denotes the current coordinates over which the filter is applied.

In terms of the Gabor filter bank, each of the individual filters seen in figure 8.1 are convolved with the leaf spots derived from chapter 7. The input spot is
shown in figure 8.8. The result of convolving this spot with the Gabor filter bank is shown in figure 8.9

Figure 8.8: A single rust spot, used for Convolution with the Gabor filter bank.

Figure 8.9: The result of convolving the Gabor filter bank with the single spot shown in figure 8.8. Note that only the real-component of the Gabor convolution is shown in this figure. The difference in sensitivity to both frequency and orientation are clearly visible. Note the high responses in the 4th and 5th images in the first (left) column. These responses correspond to the vertical transition between the light and dark tan regions in the input image. This filter response would be similar for all rust spots, while the response from a disparate infection would have an equally unique response pattern.
8.3 The Gabor Filter Usage.

8.3.2 Convolution Theorem.

This section presents an alternative to Convolution as defined above. Whilst Convolution Theorem is common in signal processing and other forms of digital image processing, the method presented here has not been explicitly reported in the literature for Gabor filters, and is considered innovative.

Despite the name similarity, the key difference between filter convolution discussed in section 8.3.1 and Convolution Theorem presented here is that the former focuses on the spatial domain, while this section focuses on the frequency domain. Despite the key differences, the two theorems are closely related.

Convolution Theorem\(^6\) states that the Fourier transform of the convolution is equal to the point-wise multiplication of the individual Fourier transforms:

\[
\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\} \tag{8.2}
\]

This relationship is significant: by taking the inverse Fourier Transform \(\mathcal{F}^{-1}\), equation 8.2 can be rewritten as:

\[
f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\} \tag{8.3}
\]

The effects of this equation are that by taking the Fourier transforms of both the input and the filter, then the frequency-domain point-wise product is directly related to spatial convolution by another Fourier transform of either side of the equation. By first converting both the filter and input into the frequency domain and then multiplying point-wise, before performing an Inverse Fourier transform to bring the filtered product back to the spatial domain, the final result is identical to the original convolution integral discussed in section 8.3.1.

The significance of this relationship is this: in terms of computing power,

\(^6\) (Press, Teukolsky, Vetterling & Flannery 1997)
8.3 The Gabor Filter Usage.

Convolution is defined as \( O(M^2, N^2) \), where \( M, N \) are respectively the numbers of elements (pixels) in the filter and input images, while the \textit{big O} notation is used in computer science to classify computing time.

Conversely, by using the Fast Fourier Transform algorithm, computational time can be reduced to as much as \( O(M\log_2 N) \). The caveat being that both \( M, N \) should be integers with a power of two.

In the present case, each disease spot was deliberately chosen to be \( 32 \times 32 \) pixels, as was the individual Gabor wavelets, giving \( M, N = 32 \times 32 = 1024 \). On a PC with a 3GHz dual core processor, convolving each of the 40 Gabor wavelets in the spatial domain resulted in a total computational time of 0.094 Seconds. Using the precomputed Fourier filter bank from figure 8.5 and Convolution Theorem resulted in a time of 0.0091 Seconds; an improvement in speed by a factor of 10. This novel approach represents a significant improvement in processing time and validates the extra steps required in using the frequency domain.

The process is explained graphically in figure 8.10.
8.4 Gabor Filter Improvements.

The Gabor filter in its basic form as described in the preceding sections gives a suitable response at higher frequencies, however at lower frequencies it can suffer from having a non-zero offset. Whilst this is fine in many instances, it usually means that the low-frequency regions (i.e. the ‘empty’ center region in figures 8.3 and 8.6) remain unfiltered by the Gabor filter bank.
The problem can best be explained by considering the 1-dimensional case. In Appendix D it was seen\(^7\) that the real and imaginary components cancel out in the negative region, leaving only the positive response. However, if the purely real curve is taken, then at lower frequencies it is possible for the Gaussian curve to overlap the axis, as shown in figure 8.11.

![Figure 8.11](image)

Figure 8.11: At lower frequencies, the inner tail of the real-component Gaussian can overlap the zero-axis. In signal analysis parlance, this results in a 'DC Offset', shown in red as the sum of the overlapping components.

The effect of this DC offset is that it makes the filter sensitive to the absolute intensity of the input at lower frequencies. The effect of this is that changes in illumination, contrast, and camera gain are reflected in the Gabor filter response. Eliminating this DC offset allows the filter to become illumination-invariant, i.e. a spot on a leaf in partial shade will return the same filter response as one in bright light.

One method of correcting this is presented in (Field 1987). By constructing the Frequency response using a logarithmic scale, the Fourier response at Frequency = 0Hz will always be nil. This is shown in figure 8.12, using the modified Log-

\(^7\)Refer figure D.10 on page 232.
Gabor frequency equation given in equation 8.4.

\[ \Phi(f) = e^{\left(-\frac{\left[\log\left(\frac{f}{f_0}\right)\right]^2}{2\left[\log\left(\frac{\sigma}{f_0}\right)\right]^2}\right)} \]  
(8.4)

Where \( f_0 \) is the center frequency of the filter, and \( \sigma \) is a shaping constant.

![Graph](image)

Figure 8.12: The Log-Gabor Frequency response, shown in both standard and logarithmic plots. By converting the Gaussian to a Log-scale, it ensures that the Fourier response is null at frequency=0 for all Log-Gabor wavelets.

One minor issue with the Log-Gabor approach though is that the spatial-filter bank can no longer be directly computed, since the equation is now discontinuous, i.e., the Log-Gabor bank must be constructed in Fourier space first and then if
required converted to the spatial domain using an inverse Fourier transform.

However this issue is not considered to be of importance here, since it was shown in section 8.3.2 that using the Fourier domain only for filtering yields improved computational performance. For comparison the Log-Gabor wavelet converted back into the spatial domain is shown in figure 8.13.

![Log-Gabor spatial response](image)

Figure 8.13: The Log-Gabor spatial response. Top is the real component, bottom the complex quadrature. Note that both are discontinuous around the principal axis, necessitating construction of the Log-Gabor filter in frequency space only. $f_0 = 0.6\text{Hz}, \sigma = 1$

Construction of the two-dimensional log filter bank follows the same pattern as outlined in section 8.2. The method of constructing 2-dimensional frequency filters is described in (Kovesi 2013). 8 orientations and 5 frequencies, with the same bandwidth/octave spacings as described above were used to generate the filter bank. The results are shown in figures 8.14 and 8.15.
The Gabor filter bank and the modified Log-Gabor bank will both be compared in chapter 11 to discern which bank gives the greatest disease recognition accuracy.

Figure 8.14: Figure shows the complete Log-Gabor Fourier filter bank.

Figure 8.15: Figure shows the effect of adding each of the Fourier filters. Compare this figure with figure 8.6. Note the increased coverage of the low frequency (center) region.
8.5 Conclusions.

This chapter explained the basic usage of the Gabor filter, and expanded the concepts into Logarithmic usage in the fourier domain. By using the frequency domain for filtering, a significant improvement in processing time was obtained.

Whilst the Gabor filter gives good results, the amount of data it produces from each disease spot quickly becomes awkward to work with, without some methods of correlating the data into a more useful dataset. This is explained in chapter 9. The actual usage of the correlated dataset in disease identification is covered in chapter 11.
Chapter 9

Gabor Data Response

Correlation.

The Gabor filter derived in the previous chapter necessarily produces a very large amount of data: for a $32 \times 32$ input image with 40 wavelets in the filter bank, the Gabor filter response is $(32 \times 32) \times 40$ with real and complex components. This means a total of 81,920 data-points is generated per leaf spot.

As outlined in the literature review, the problem of reducing the size of the data is non-trivial, however the methods are only briefly discussed, if at all. The lack of ‘tried and true’ established methodologies thus leaves this section open to interpretation. As such this chapter presents an explanation of three methods, while later chapters focus on a direct evaluation of each within the complete system to establish which gives the best results on the actual dataset.
9.1 Data Presentation.

9.1.1 Preliminary Stages.

As an initial step in data correlation, some consideration is needed on how to best approach the complex nature of the data. The literature proposes two alternate methods for a first-step reduction in data:

1. To retain the real-value component filter response, and discard the complex component (Chao 2010).

2. Attempt to combine both components into a meaningful subset.

Of these options, the first is self-evident, while the second requires elaboration. As the data is complex, it has both a phase angle and an absolute length (magnitude) of the response amplitude.

Where:

\[ \sqrt{real^2 + complex^2} \]

Is the magnitude, computed per-pixel pair, and

\[ \tan^{-1} \frac{Complex}{Real} \]

Gives the phase angle.

The magnitude and phase angle are compared graphically in figure 9.1. This comparison yields interesting results. Namely, the relative illumination response across the image is contained in the amplitude component, along with imaging contrast and camera gain, while the phase component is unaffected by lighting variations. It is for this reason that Daugman (2004) chooses to use the magnitude component\(^1\). Both magnitude and phase responses will be analysed in chapter 11.

\(^1\)Daugman (2004) further reduces the data by quantizing the quadrant of the phase response, giving only one of four possible values for each pixel and hence two bits of phase information. This was done due to the limited storage available on magnetic identity cards, which was the application discussed in the citation.
in order to discern which is the better classifier.

Whilst taking either the magnitude or phase response reduces the available data by half, this still presents a very large subset, which needs to be reduced further for practical analysis. Three approaches will be discussed in subsequent sections.

Figure 9.1: Top row: left is the Real (cosine) component of the Gabor filter bank. Right is the Complex quadrature (sine) component. Bottom row: On the left is the magnitude of the real and complex components, while on the right is the phase angle of the real and complex components. From this it is evident that the illumination (contrast) is contained entirely in the complex magnitude component, while the phase component is unaffected by contrast variations in either the filter wavelets themselves as shown here, nor in the filter response images.
9.2 **Principal Components Analysis.**

The first method in data correlation to be examined focuses on Principal Component Analysis (PCA). From Smith (2002), “*PCA is a useful statistical technique with many applications in pattern finding in data analysis, as well as facial recognition and image compression in machine vision applications.*”

At its core, the concept of PCA involves finding correlations between two or more datasets, by identifying (if possible) an orthogonal transformation which converts a dataset into a second set of values of linearly uncorrelated variables called *principal components*. By finding patterns in data, it is possible to establish correlations between the data, as well as to reduce the dimensionality of the dataset, as applicable here.

Two prominent methods for PCA exist; the first uses the Eigenvectors approach, while the second uses Single Value Decomposition (SVD) (Shlens 2003). Both approaches will be compared at the end of this section, with Appendix E comparing the methods in detail.

In order to explain the process of dimensionality reduction, it is useful at this stage to include a visual example. Consider\(^2\) the leaf shown in figure 9.2. In the image, the leaf is oriented at an angle with respect to the blue XYZ Cartesian coordinate system.

---

\(^2\)All CGI artwork in this chapter generated by the author.
9.2 Principal Components Analysis.

The leaf itself exists in three dimensions: length, width, and depth, in descending order of size (i.e., the leaf is longer than it is wide). It may however be necessary to condense the dimensionality of the leaf, such that it can be displayed in a 2-dimensional coordinate system.

An obvious way to display the leaf in two dimensions would be to preserve the length and width of the leaf, and discard the depth information. The question is, given the leaf in the blue XYZ coordinate system, how to identify the principal dimensions. This is where Principal Components Analysis comes in.

Figure 9.3 shows a ‘point cloud’, containing discrete nodes representing points on the surface of the leaf. Each one of these points has its own XYZ coordinate location in space.
Figure 9.3: The purple dots denote a ‘point cloud’, containing 500 discrete nodes on the surface of the leaf. PCA uses the XYZ coordinates of each of these nodes to find the principal axes.

The core precept of Principal Components Analysis is that it is possible to mathematically correlate the coordinates of each of the 500 points contained in figure 9.3 to establish patterns of dimensionality. This correlation can be achieved by either taking the Eigenvectors of the dataset, or by Singular Value Decomposition. Both these methods are discussed in detail in Appendix E, with a brief comparison of the methods given as:

- The Eigenvalues method computes the covariance matrix of the data. That is, it establishes how much the values in one axis change with respect to another axis. The Eigenvectors are then computed for this covariance matrix. The Eigenvectors give a new coordinate system which matches the data, with the corresponding Eigenvalues denoting the order of importance of the new Eigenvector axes. By transforming the coordinates of each point to match the new coordinate system, the axes corresponding to low Eigenvalues can be discarded.

- SVD also uses the covariance matrix, but according to (Shlens 2003),
9.2 Principal Components Analysis.

provides a more general method for the change of axis.

The result of performing PCA on the point-cloud is shown in figure 9.4. The red $X'Y'Z'$ coordinate system aligns with the principal components of the leaf. That is, The $X'$ axis represents the ‘line of best fit’ along the longest axis of the leaf, the $Y'$ axis aligns with the leaf width, while the $Z'$ axis is aligned with the least valuable thickness (depth) axis.

![Figure 9.4: PCA on the 500 purple nodes The new red $X'Y'Z'$ coordinate system both aligns with the ‘line of best fit’ through the purple nodes, and is also located at the centroid of the leaf.](image)

The significance of this change of coordinate system is that it mathematically locates the most valuable axes. In figure 9.4, the $X'$ and $Y'$ axes contain respectively the length and the width of the leaf. The variation in ‘depth’ in the $Z'$ axis can be thought of as the least valuable axis of this image.

The concept of least valuable is illustrated in figure 9.5. In this figure, the image has been rotationally transformed so that the length and width ($X'$-$Y'$) of the leaf align with the original blue X-Y plane. The data contained in the $Z'$ axis has been completely discarded so that this image only contains the length and
width data. This final image shows how by transforming the coordinates, it is possible to discard the least valuable axes with only a minimum of data loss, as the principal components are preserved.

Figure 9.5: Result of transforming the coordinate system from the ‘global’ to the PCA coordinates. The Z’ axis has been discarded, meaning that the whole leaf can be displayed in 2 dimension (i.e., X’Y’.) Had we simply taken the original global XY axes and discarded the global Z, much of the length data of the leaf would have been lost. Thus, PCA allows data compression, by first identifying the valuable data to be preserved.

In the chapter introduction it was noted that the Gabor filter response produces a very large dataset. Even with taking the phase or magnitude, this still results in some 40,960 data points generated.

The first step to reducing this data is to combine all 40 filter responses into one dataset. By taking each 32 × 32 filter response and ‘stacking’ them, a 32 × 32 × 40 dataset is generated. This new dataset exists in 40th dimensional space and thus cannot be visually represented. However by performing PCA on this dataset it is possible to reduce this dimensionality down as far as perhaps 2 or 3 dimensions, depending on the computed Eigenvalues.

The first step in this data reduction process is to reshape each of the 32 × 32 pixel images into single 1024 × 1 feature vectors. For each layer (dimension) in
9.2 Principal Components Analysis.

The dataset, this can be accomplished as follows:

\[
\begin{array}{cccc}
X_1Y_1 & X_2Y_1 & \ldots & X_{32}Y_1 \\
X_1Y_2 & X_2Y_2 & \ldots & X_{32}Y_2 \\
\vdots & \vdots & \ddots & \vdots \\
X_1Y_{32} & X_2Y_{32} & \ldots & X_{32}Y_{32}
\end{array}
\Rightarrow
\begin{array}{c}
X_1Y_1 \\
X_1Y_2 \\
\vdots \\
X_1Y_{32} \\
X_2Y_1 \\
\vdots \\
X_{32}Y_{32}
\end{array}
\]

The result of continuing this process over all 40 dimensions is that a 1024 \times 1 \times 40 feature vector is produced. From this a 40 \times 40 covariance matrix is generated, and the 40 Eigenvectors computed, corresponding to a rotation of axes to find the ‘lines of best fit’ across 40th dimensional space.

While it isn’t possible to visualise a 40th dimensional dataset, the reshaping and PCA process is explained for the Gabor filter responses in the following images.

Figure 9.6 Shows the first stage of ‘stacking’ each of the images to generate a 40th dimensional dataset.
Figure 9.6: The 40 Gabor $22 \times 32$ pixel filter responses are ‘stacked’ into a single 40th dimensional dataset.

Figure 9.7 Shows the result of reshaping each dimension into $1024 \times 1$ vector. Repeating this process over all 40 dimensions results in a $1024 \times 1 \times 40$ feature vector. The covariance matrix would be generated at this stage, and the Eigenvectors and Eigenvalues computed. Reshaping the dataset to the new coordinate system, and taking the two most important Eigenvectors is shown in figure 9.8.
9.2 Principal Components Analysis.

Figure 9.7: Each of the 40 Gabor responses are reshaped from $32 \times 32$ pixel images into $1 \times 1024$ vectors, giving a total $1024 \times 1 \times 40$ feature vector.

Figure 9.8: After the principal components is computed on the $1024 \times 1 \times 40$ feature vector, the data is transformed to the new principal axes. By analysing the Eigenvalues, the low-value dimensions can be discarded, leaving just two dimensions.
Figure 9.9 shows the result of reshaping the two salient PCA vectors back into $32 \times 32$ pixel images. The ‘ghostly’ xray-like images represent each of the high value points across the original 40 filter response images. These two images indicate that the entire 40 image spots contains a very large amount of redundant data. By identifying the salient datapoints this redundant data can be largely discarded, with only a minimal loss of data.

Figure 9.9: The two PCA vectors are reshaped into $32 \times 32$ pixel images. Each of the white spots in the images corresponds to high-correlation points across the original 40 filter images. Note that this reshaping step is shown for visual purposes only; the subsequent neural network stage discussed in the next chapter requires the data to be in an $X \times 1$ vector.
9.2 Principal Components Analysis.

9.2.1 PCA - Comparison and Conclusions.

Comparing Eigenvalues and Singular Value Decomposition in Appendices E.1 and E.2 yielded interesting results. Shlens (2003) indicated that SVD provides a more general method of change of axis, however it was anticipated that the extra steps in the SVD approach to compute $U$ would result in increased computing overhead. To discern if SVD would indeed take longer to process, it was necessary to provide a side by side comparison of methods, using both the $10 \times 2$ example matrix from Appendix E, and the actual $1024 \times 1 \times 40$ Gabor spot matrix$^3$.

The $10 \times 2$ test matrix and $1024 \times 1 \times 40$ Gabor spot matrix were analysed using both the Eigenvalue approach, and the SVD Method. Due to the way the inbuilt SVD command in Matlab functions, there is no way to manually select which of the components to compute. As this results in a waste of processing time generating unnecessary arrays, it was decided to manually compute the matrix $Y^T Y$, as well to provide a better comparison. Comparisons of the three methods are shown in table 9.1: Eigenvalues, SVD, and manual computation of the $Y^T Y$ component are performed, respectively on both the $10 \times 2$ example and $1024 \times 1 \times 40$ Gabor datasets.

Table 9.1: Summary of comparison between Principal components methods. Times were developed using Matlab on a PC with 3Ghz Processor, 2GB RAM. The top two rows give a direct comparison of Convolution techniques, while the lower three give times for Eigenvector computations.

<table>
<thead>
<tr>
<th></th>
<th>$10 \times 2$ Example Dataset</th>
<th>$1024 \times 40$ Gabor Filter Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigenvalues Convolution Matrix</td>
<td>87$\mu$s</td>
<td>1.10ms</td>
</tr>
<tr>
<td>PCA Convolution vector ‘Y’</td>
<td>44$\mu$s</td>
<td>0.8ms</td>
</tr>
<tr>
<td>Eigenvectors of Convolution Matrix</td>
<td>55$\mu$s</td>
<td>0.71ms</td>
</tr>
<tr>
<td>Singular Value Decomposition of ‘Y’</td>
<td>85$\mu$s</td>
<td>221ms</td>
</tr>
<tr>
<td>Eigenvectors of $Y^T Y$</td>
<td>45$\mu$s</td>
<td>1.1ms</td>
</tr>
</tbody>
</table>

$^3$The dataset from figure 8.9 on page 96.
From the results of table 9.1, it is evident that with smaller datasets, most of the processing time is taken up actually computing the convolution matrix, with the Eigenvalues approach explained in section E.1 producing a total time of $87\mu s + 55\mu s = 142\mu s$ for the combined convolution and Eigenvectors steps. By comparison the SVD approach given in section E.2 gave a time of $129\mu s$, a slight improvement due to the time required to produce the convolution matrix. Separately computing $Y$ first, and then just taking the Eigenvectors and values gave $89\mu s$, indicating that with smaller datasets, using this manual ‘combination’ approach gives improved results.

At larger datasets however, it is evident that relying on SVD produces a marked increase in processing time, due to the algorithm’s need to compute the Eigenvalues of a $1024 \times 1024$ matrix, which is subsequently not used in finding the principal components. Whilst there is still a minor improvement in time by computing $Y$ separately, this is subsequently lost by manually computing the Eigenvectors of $Y^T Y$.

In conclusion, as this process must be computed for each spot on each leaf, any savings in processing time are important. Computing the Eigenvectors of $Y^T Y$ approach offers no advantage over the method defined in section E.1, thus the Eigenvectors approach is the method which will be used for subsequent sections.

Having established the most efficient method of computing the Principal Components of a large dataset, all that remains is to analyse the correlation of multiple Gabor filter responses, and compare their Eigenvalues to determine the size of the feature vector for use in the Artificial Neural Network, covered in chapter 10.

Applying the Eigenvectors approach to multiple $1024 \times 40$ Gabor filter responses spots yielded a $40 \times 1$ matrix of Eigenvalues for each spot. Looking at the first five Eigenvalues of three sample rust spots gave the following:

\[\text{The approach and timing of each was directly computed in Matlab. This may indicate an inefficiency in Matlab's 'cov' command for producing covariance matrices, as this process is fundamentally identical to producing the matrix } Y^T Y.\]
9.2 Principal Components Analysis.

<table>
<thead>
<tr>
<th></th>
<th>spot1</th>
<th>spot2</th>
<th>spot3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.419</td>
<td>0.101</td>
<td>1.146</td>
<td></td>
</tr>
<tr>
<td>0.136</td>
<td>0.063</td>
<td>0.892</td>
<td></td>
</tr>
<tr>
<td>0.063</td>
<td>0.048</td>
<td>0.207</td>
<td></td>
</tr>
<tr>
<td>0.053</td>
<td>0.029</td>
<td>0.160</td>
<td></td>
</tr>
<tr>
<td>0.035</td>
<td>0.024</td>
<td>0.086</td>
<td></td>
</tr>
</tbody>
</table>

$E_{igenvalues(rust)} =$

From the Eigenvalues, it is evident that the highest-value principal components of the Gabor filter response in each case are the top one or two. This means that the dimensionality of the entire response can be compressed down as low as one or two dimensions total. In order to avoid over-compressing the dataset, the top five principal components were taken for analysis, as reintroducing some redundancy into the Artificial Neural Networks prevents helps to prevent over-fitting during the training stage. Training of Neural Networks is covered in chapter 10, with specific analysis and validation of the PCA methods covered in chapter 11.

As a purely graphical representation, the rust spot and Gabor filter response shown in figures 8.8 and 8.9 on page 96 had its Principal Components computed as per the above process, and the resulting $1024 \times 2$ feature vector reshaped into $32 \times 32$ images representing the first two PCA responses. These images are shown in figure 9.10. These images effectively represent only those pixels from the filter response which have a high degree of redundancy across all 40 individual filter responses. It is interesting to consider the results of the image: ideally these two images represent the salient pixels for the entire filter response of the spot, and should give a unique result for any spots of this type of disease. The increased contrast in the left image is also indicative that this image corresponds to the highest-value principal component.

Note that this graphical representation is purely to highlight the results of using PCA to reduce the dimensionality down from 40 to 1 or 2: under normal circumstances the data would be left as an $n \times 1$ feature vector for subsequent analysis steps.
Figure 9.10: The first two principal component ‘Eigenspots’ of the rust spot shown in figure 8.8. Note this representation is for graphical purposes only. The principal eigenvector (i.e. the principal component corresponding to the largest Eigenvalue) is on the left.
9.3 Gray Level Co-occurrence Matrix.

The previous section presented the PCA method for reducing the dimensionality of the data-set. This section presents an alternative approach, which will also be examined in later chapters to see if it offers an improvement over Principal Components.

In his source, Bau (2008) provides a different approach to data compression, by taking the Gray Level Co-occurrence Matrix (GLCM) of the Gabor response, and using this to compute the total energy, correlation, contrast and homogeneity of the data-set.

GLCM works by considering the co-occurrence between adjacent pixels. Each time that a pixel value appears adjacent to another one, the count is recorded. Considering a $32 \times 32$ image consisting of a uniform colour where every pixel in the image has the same numeric value, the total co-occurrence matrix would record this single adjacent value 1024 times.

The process is better explained with a more useful example than a single uniform image. Consider a $4 \times 4$ pixel image, consisting of pixel values between 0 and 3, as shown:

$$
\begin{array}{cccc}
0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 \\
0 & 2 & 2 & 2 \\
2 & 2 & 3 & 3 \\
\end{array}
$$

In the top row, 0 appears next to 0 once, 0 next to 1 once, and 1 next to 1 once. Continuing in this fashion, a GLCM can be constructed, listing each value according to its horizontal neighbour. The complete GLCM for the image data is shown below, with the numbers above and to the left of the main $4 \times 4$ matrix indicating the neighbouring left/right pixel values. It is important to note that

---

5This dataset is adapted from (Hall-Beyer 2007).
the GLCM should be symmetrical so that the ‘left’ neighbour count must match the ‘right’ neighbour count, i.e., the number of times that 0 next to 1 in the first row should equal the number of times 1 is next to 0.

\[
GLCM = \begin{bmatrix}
0 & 1 & 2 & 3 \\
0 & 4 & 2 & 1 & 0 \\
1 & 2 & 4 & 0 & 0 \\
2 & 1 & 0 & 6 & 1 \\
3 & 0 & 0 & 1 & 2
\end{bmatrix}
\]

In the GLCM, the top row reads 4,2,1,0. This means that in the image data-set, 0 appeared next to 0 four times horizontally (reading both left and right) and so forth through the whole image. 2 appeared next to 2 six times, etc. The main diagonal indicates how many times a pixel was adjacent to another pixel with the same value. Thus the main diagonal is a good measure of how uniform the image is.

The true power of the GLCM comes from the subsequent steps\(^6\): once constructed, the Energy in the original image can be simply computed by taking the sum of the squared elements in the GLCM. High energy levels occur when the image is very orderly.

\[
Energy = \sum_{i,j} p(i,j)^2 
\]

(9.1)

where \(p(i,j)\) is a normalising equation, based on dividing each value in the GLCM by the the sum of all values in the cell\(^7\).

In a similar manner, the Contrast of the original image is computed from the GLCM. This gives a measure of the intensity contrast between each pixel and its neighbour.

\[
Contrast = \sum_{i,j} |i - j|^2 p(i,j)
\]

(9.2)

---

\(^6\)From (Hall-Beyer 2007) and (MathWorks, Inc. The 2010).

\(^7\)Hall-Beyer (2007) treats the normalising equation as value as the ‘probability’ of each value occurring in the GLCM, and is given as: “the number of times (an) outcome occurs, divided by the total number of possible outcomes.”
Correlation is a measure of the probability of occurrence of pixel-pairs of a specified grey level, computed over the whole image. Correlation will be high for repeating patterns:

\[
\text{Correlation} = \sum_{i,j} \frac{(i - \mu_i)(j - \mu_j)p(i,j)}{\sigma_i \sigma_j}
\]  

(9.3)

Homogeneity measures the closeness of the distribution of elements in the GLCM to the main diagonal. Since the GLCM diagonal represents pixel-pairs with the same value (i.e., 1 next to 1) a high homogeneity indicates a large number of similar pixels:

\[
\text{Homogeneity} = \sum_{i,j} \frac{p(i,j)}{1 + |i - j|}
\]  

(9.4)

From the above four conditions, it is possible to generate unique texture feature data from each of the 40 filter response, with the overall pattern of each of the four responses providing a unique identifier for each of the filter responses.

In the above example however, only the horizontal pixel correspondence was considered. For the system to truly return the values of the Gabor responses, it too must vary in both frequency and orientation. Fortunately, in this instance, frequency can be simply defined as the distance between each pixel considered during co-occurrence: in the above example this was each pixels immediate neighbour. In actual practice however, there is often not a lot of variation between directly neighbouring pixels. By visually inspecting the results from figure 8.9, taking every 2nd and every 4th neighbour provides the greatest coverage, across each of the \(32 \times 32\) pixel responses.

Likewise, in the above example only horizontal co-occurrence was considered. By taking the second and fourth pixel, at each of the horizontal, vertical, and two diagonal (i.e. \(0^\circ, 45^\circ, 90^\circ, 135^\circ\)) co-occurrences, equations 9.1 to 9.4 can be computed for each to give the best sampling of the total Gabor filter response.
In summary, each of the 40 Gabor filter responses has Energy, Correlation, Contrast and Homogeneity computed across four rotations, and two pixel spacings. By preserving the order of the 40 filters (i.e. the original $5 \times 8$ wavelet array) each spot produces a unique feature vector, consisting of $40 \times 4 \times 2 \times 4 = 1280$ values. As each spot of each disease should have both similar Gabor responses, and similar GLCM energy responses, it is proposed that this approach will give a fast and accurate texture classifier. This approach will be discussed further, and validated on an actual dataset in chapter 11.
9.4 Conclusions.

This chapter described three methods for data correlation, with two different approaches for Principal Components Analysis. PCA analytically samples the data to find the most valuable responses to the Gabor data, while GLCM directly samples the energy and other correspondences across each filter response, to statistically deduce texture features.

Two Principal Components methods were analysed for speed, with the conclusion being that computing the Eigenvectors required less processing time to analyse than Singular Value Decomposition.

Both PCA and GLCM methods presented in this chapter reduce the size of the data, using different approaches. Further comparison of the methods to discern which gives the best response requires testing with the complete system. This is covered in chapter 11.
Chapter 10

Artificial Neural Networks.

In the previous chapter, it was shown that the data-set produced by the Gabor filter could be correlated into a much smaller subset. The use of this subset to develop a classifier using the Artificial Neural Network (ANN) will now be discussed.

ANN learning represents one of the basic tenets of Artificial Intelligence. From (Burger 1996), ANNs are processing devices that are loosely modelled after the neural structure of the mammalian cerebral cortex but on much smaller scales. The simplest definition of an ANN can be attributed to Dr. Robert Hecht-Nielsen, the inventor of one of the first neuro-computers. Dr Hecht-Nielsen defines neural networks as: “...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs.”

Since the inception of the ANN in the 1950’s and 60’s, the concept has continued to be expanded on and developed to the point where they are proficient classifiers and in particular, are well suited to addressing non-linear problems and other related machine learning applications. By presenting the ANN with a training set of known samples and instructing it to iteratively tune itself until the network produces the desired outputs on the known samples, the system will be able to recognise unknown samples of the same diseases that it has been trained with.
10.1 Artificial Neural Network Outline.

This is the basic concept behind supervised machine learning.

10.1 Artificial Neural Network Outline.

The basic premise of the ANN is based around the concept of a perceptron. Figure 10.1 shows the basic structure and function: each of the inputs into the perceptron are weighted and summed, and if this exceeds the threshold then the output of the perceptron is triggered.

\[ \sum \theta \]

Figure 10.1: Basic Perceptron arrangement. If the sum of the inputs exceeds the threshold value then the output is triggered ‘high’. Not shown is the weighting; each of the inputs is typically given a weighted value, the value of which is determined programmatically by the software. On the right is the graph of the output.

In practice, the threshold value and individual input weights from figure 10.1 are determined analytically by the software itself. A sample input of data (in this case, the filter outputs from chapters 8 and 9) is fed into the neural network with a known output (i.e., the known disease) and the system iteratively tuned until the weights give the desired output for the known input. Typically this is done with a large number of training samples for each disease that the system is being trained for, to account for any visual variations in the infection such as age, size and spot connectedness. To prevent over-fitting\(^1\), it is also necessary to include a number of false samples, in the form of either misdiagnosed spots (see chapter 7) or healthy swatches of leaf.

---

\(^1\)Over-fitting in this case can best be thought of as making the system too well tuned for a specific dataset, i.e., it could be trained to a very high precision on the training samples themselves, but will not have learned to generalise to recognise new samples.
While the basic perceptron concept is simple to understand, using it in a practical application is less so. A full Neural Network structure consists of numerous perceptrons in multiple layers, and the substitution of a sigmoid transfer function in place of the Heaviside trigger. This latter modification\textsuperscript{2} to the system makes it ‘less hard’, and thus allows it to ‘feel’ the data more precisely; this permits the rate of change of each neuron to be communicated to the other layers incrementally. The system also requires a modification to allow feature data from higher levels to adjust the lower: if all the weights in the multi-layer perceptron are potentially changeable, the information reaching a particular layer cannot be relied upon in a purely feed-forward system.

The altered back-propagation system then functions as follows:

1. The input is forward propagated through the system in order to generate the propagation’s output activations.

2. The system’s output activations are then back-propagated through the system using the training pattern target so as to generate the deltas (rate of change) of all output and hidden neurons.

This process is repeated recursively until the overall rate of change of the system between each iteration has reached some predefined equilibrium threshold. Once the system is trained, with all weights and threshold values established, it then functions in a purely feed-forward system for any unknown datasets it is expected to analyse. Figure 10.2 shows the modified single perceptron with back-propagation used to correct the weighting.

\textsuperscript{2}From (Davies 2012).
Figure 10.2: Basic back-propagation perceptron. Initially, weighting $W_i$ is given a random value to start the system running. At each iteration of data through the perceptron, the computed output is compared to the desired output, and the correction factor $\beta EX/|X|$ adjusted to suit.

The reader is encouraged to investigate a source such as Davies (2012) for further information on multi-layer feed-forward back-propagation Neural Networks. For now it is sufficient to consider figure 10.3 which represents a full network.

Figure 10.3: The complete feed-forward Back-propagation Neural Network. In this case, there are two hidden layers, with data flowing from left to right. The threshold values and weightings are hidden for clarity. The number of layers, and the number of nodes contained within varies depending on the complexity of the system being analysed and is discussed in a separate section. On the right is the modified sigmoid triggering function.
10.2 Network Structure.

From figure 10.3, it is evident that the system can have multiple layers, with multiple (often many) neurons per layer. Adding extra layers has the effect of making the training result non-linear, while having more neurons than necessary effectively turns the system into a memory bank that can recall its training set perfectly, but which performs badly when presented with samples that weren’t part of the original training set\(^3\).

10.2.1 The Input Layer.

Each system has a single input layer. The number of nodes contained in the input layer is dependent on the structure of the dataset. From chapter 9, each disease spot consisted of a 5120 (for 5 layers of PCA) or 1280 (GLCM) data-point feature vector. This means either 5120 or 1280 nodes in the ANN input layer.

The Training Input.

The system must be trained before use. The training set consists of \(X\) number of disease samples, and so the input layer during the training stage will contain \(5120 \times X\) entries. When presented with a single unknown sample after training, the input will simply be \(5120 \times 1\).

---

\(^3\)This is the main cause of over-fitting, and can be thought of as not having enough data present to train the excess number of neurons. The system learns the training set perfectly, but is unable to recognise any unknown images it is presented with. Showing a child who has never seen a dog before ten poodles, and then asking if a greyhound is also a dog is a good example of training a complex network (the brain in this case) on insufficient data.
10.2 Network Structure.

The Validation Input.

The during training, the input layer $X$ is usually broken up into training, validation and test samples, with random samples taken from the main dataset for each. While the Training input is actually used for tuning the network, at each back-propagation iteration the weights of each node is necessarily altered, in turn changing the way that the training samples ‘move through’ the network. Because of this, a separate (usually smaller) number of validation samples is fed through the network between the end of one back-propagation phase and the beginning of the next feed-forward iteration. This data thus remains unaltered by the back-propagation, and allows the system to genuinely gauge performance on unaltered data. Functionally this relationship means that if the training input increases, while the validation input decreases, then the system is over-fitting on the training inputs.

The Testing Input.

The final sample to be taken from the input layer is the testing set. This represents a dataset that the network has had no knowledge of during the training and validation stages. This is equivalent to feeding actual samples into a trained network, and gives a final indicator of how well the system performs. If all three sets show a similar value of accuracy at the end of the training phase, then it is a good indicator that the system is correctly fitted.

10.2.2 The Output Layer.

Each system has a single output layer. The number of nodes in the output layer consists of the desired number of diseases that the system is being trained for.
The training Output.

During the training stage for three diseases, the output dataset is $3 \times X$, where $X$ is the number of samples, and matches the same number of samples as used for the input training set. The structure of this dataset is that each disease is assigned a row in the matrix, with a 1 in the row corresponding to the ‘correct’ disease for each sample, and zeroes elsewhere. Each column entry in the output matrix matches the corresponding column in the input matrix: in this fashion the ANN can relate each input sample to its known output on a per-column basis.

10.2.3 The Hidden Layers.

Research into an optimal figure for both layers and neurons indicates that it is a fairly empiric process. Davies (2012) indicates that no more than two hidden layers should ever be necessary, and that frequently training one layer ‘through’ another results in uncertainty in the system and increased training times. (Heaton 2008) suggests that two layers are rarely needed, and that “one layer can approximate any function that contains a continuous mapping from one finite space to another”, while two layers “can represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy.”

Since using ANN for disease detection is effectively a continuous problem (i.e., each spot on the leaf can only be classified as one disease) then using a single hidden layer seemed appropriate. However, as the Gabor response data requires a greater degree of ‘intuition’ to detect patterns, the testing phase quickly showed that two layers gave superior performance. A single layer will not be discussed further: system evaluation in chapter 11 focuses on only two layer networks.

The number of neurons in each layer is more critical. Too few neurons will result in under-fitting, whereby the system is unable to detect the patterns in a complicated dataset. Similarly, too many neurons results in over-fitting, whereby
the system has far more processing power than necessary. In this situation, the
dataset is effectively too small to adequately train each of the neurons.

From Heaton (2008), there are numerous schools of thought as to how many
neurons to use:

1. The number of neurons in each hidden layer should be between the values
   used for the input and output layers

2. The number of neurons in each hidden layer should be roughly 2/3 the
   number of input neurons, plus the size of the output layer.

3. The number of hidden neurons should be less than twice the size of the
   input layer.

4. The number of neurons should be the average value of the input + output
   neurons.

The above is often treated as a guide, with the final value found by trial and
error. The actual network architecture is derived in chapter 11, during final
system evaluation.
Chapter 11

Evaluation of the System.

This evaluation chapter is used to ‘bring together’ all the previous chapters, and to validate the various results from each which were left outstanding in the preceding pages. Figure 11.1 shows the concept flowchart for this chapter, outlining the remaining stages to be completed before the final system design can be chosen.
Figure 11.1: The flowchart shows the chosen alternate processes for each stage. Outstanding options will be resolved in this chapter.
This chapter is divided into two sections. Initial analysis and validation will focus on disease samples provided by Dr Steven Neate. These samples will be used to establish which of the flowchart approaches works best on actual disease datasets. The remaining stages to be confirmed will be analysed using a full work-flow approach. i.e., each remaining branch of the flowchart will be tested to select the best of the following branches:

1. From chapter 8, Both the Gabor and Log-Gabor filtering methods will be separately evaluated for performance.

2. The Principal Components Analysis from chapter 9 will be evaluated, as will Gray Level Co-occurrence Matrix methods. Using either the magnitude or phase angle of the complex filter responses will also be analysed to discern which gives a better performance.

3. The structure of the Artificial Neural Network from chapter 10 will be optimised for the data, and trained to recognise the three diseases: Mildew, Tan spot, and Stem Rust as identified in chapter 2.

Once the Gabor filter, Data Correlation and Neural network approaches have been chosen based on the disease datasets, analysis of actual photographs will be used for final system validation. This validation stage will also include a detailed comparison of both HDR compositing on a high-quality SLR camera, and colour compositing methods using embedded cameras.
11.1 Analysis of existing images.

In order to evaluate the system it was first deemed necessary to establish a baseline to compare the Gabor filter responses to and also to train an initial, simpler neural network. Instead of using the Gabor filters and subsequent data correlation, the baseline simply involved taking the disease spot samples themselves, and using directly in the Neural Network. The process was as follows.

The image samples supplied by Dr Neate, and from (DAFF 2012) were initially cropped in photo editing software\(^1\) to maximise the amount of leaf per frame and eliminate as much of the background as possible. The disease spots were then extracted using K-means clustering, as per chapter 7. The results of this on one leaf are shown in figure 11.2. Repeating this process for each sample yielded a total of:

- 89 Stem Rust samples.
- 119 Tan Spot samples.
- 96 Mildew samples.

The above sample set was augmented with 259 samples of healthy leaf swatches\(^2\). Including non-diseased samples prevents over-fitting of the Neural Network. This brings the total number of spots for training to establish a baseline to 563 samples.

\(^1\)As the sample images were supplied by Dr Neate, automatic depth methods and the use of a gray reference card weren’t possible. The leaves in these samples were simply extracted using photo-editing software.

\(^2\)This number was derived by first subtracting the diseased regions from the leaf, and then dividing the remaining region of the leaf into 32 × 32 pixel swatches.
11.1 Analysis of existing images.

Figure 11.2: The leaf was manually cropped in photo editing software to fill the screen as best as possible, and the K-means extraction algorithms from chapter 7 were used to extract the diseased spots. This process was repeated for each leaf to build up a disease data-bank.

With the RGB data-bank of extracted samples created, each of the spots was converted into HSV colour space, and the Hue channel reshaped from a $32 \times 32$ pixel image into a $1024 \times 1$ feature vector. All the samples were added as columns in a training set, giving a $1024 \times 563$ matrix. This matrix became the input layer for the Neural Network. As there were three diseases, plus the healthy swatch data-set, the output layer consisted of a $4 \times 563$ matrix. Respectively, each row of this output matrix represents Stem Rust, Tan Spot, Mildew, and Healthy swatches. Each column in the output matrix corresponds to the matching column in the input matrix, giving the ANN the ‘correct’ value to tune to for each input sample. Figure 11.3 shows a representative sample from each of these spots. It is clear that even at a size of $32 \times 32$ pixels, it is still possible to visually differentiate between each disease.
11.1 Analysis of existing images.

Figure 11.3: Left to right: Rust, Tan spot, Mildew, Healthy leaf. Even at $32 \times 32$ pixels, the samples are visually dissimilar enough to make classification possible.

From chapter 10, a neural network with two hidden layers was constructed. The number of neurons was initially chosen as the average of the input plus the output layers. This value was empirically adjusted to give 200 neurons in the first hidden layer, and 100 in the second. The total neural network structure is shown in figure 11.4.

![Artificial Neural Network back-propagation training structure with two hidden layers and sigmoid triggering, programmed with the standard Matlab Neural Network algorithms.](image)

This system was trained using Matlab’s inbuilt Neural Network algorithms. The software automatically partitioned the input data into training, test and validation datasets, with 395 randomly selected samples used for training, 84 samples for testing and 81 for self-validation. The system was trained using back-propagation, with the delta function (rate of change) of the system analysed at each iteration, until the system validation reached its lowest Mean Squared Error (MSE), this occurred after 114 epochs\(^3\). The results of this training performance is

\(^3\)An epoch represents each iteration the network is presented with a new input pattern via back-propagation.
shown in figure 11.5, with the rate of change indicating a good steady convergence towards the lowest MSE value.

**Best Validation Performance is 0.02628 at epoch 114**

![Graph showing system convergence](image)

**Figure 11.5:** Shows the system convergence at epoch 114.

The training confusion matrix for the system is shown in figure 11.6. Here the green cells in each of the four matrices show the correct diagnosis for each of the inputs, while the red cells show misclassified samples, based on the ‘correct’ output for each disease. The lower left ‘test’ plot is the most relevant in this situation. After the system is tuned on the training and validation datasets, the test set is presented to the fully trained network, in order to gauge how accurate the system diagnosis is. On this dataset over 90% of the disease spots were correctly identified. This number will be the value for the Gabor/Log-Gabor filtering system to improve on in the subsequent sections of this chapter.
11.1 Analysis of existing images.  

Figure 11.6: The system confusion matrix. The green cells on the diagonals show the correctly classified samples for each of the test, training and validation datasets. The red values show incorrectly classified samples. These matrices are interpreted as follows: focusing on the Test Confusion Matrix, there are four incorrect samples for target class 3 (below the green 11 on the center diagonal). This four corresponds with Target Class 3, and Output class 4. This means that these four samples from class 3 (i.e., Mildew) were misclassified as healthy samples (class 4). As there was 84 samples total used in testing, this misclassification represents an error of 4.8%.
11.2 Principal Components Analysis.

This section tested the Gabor and Log-Gabor filtering methods for validity. Both Phase and Magnitude were sub-sampled using PCA, with the resultant data subsets used to train appropriate Neural Networks. Results are presented below, with separate subsections describing each alternative approach.

11.2.1 Analysis of the Gabor Filter.

This section focussed on using the Gabor wavelets to filter the disease spot images, before sub-sampling with PCA and training the Neural Network. The testing process was similar to that defined in the preceding section:

1. Each of the sample spots from the preceding section (i.e., 89 Stem Rust, 119 Tan Spot, 96 Mildew, and 259 Healthy samples) were loaded into the software.

2. Each of the samples was converted to HSV colour space.

3. The Hue channel from each sample was converted to frequency space using the Fast Fourier Transform\(^4\).

4. The frequency spots were then multiplied by each of the 40 Gabor filter frequency wavelets\(^5\).

5. Each of the filtered spots was then converted back into the spatial domain by using the Inverse Fourier transform.

6. The Magnitude, and Phase angle of each complex-filtered pixel was then computed, in order to independently evaluate which method gave the better results.

\(^4\)Refer (James, Burley, Clements, Dyke, Searl, Steele & Wright 2011) for information on the Fast Fourier Transform.

\(^5\)See figure 8.5 on page 93.
11.2 Principal Components Analysis.

The filter responses were then correlated using Principal Components Analysis, as per chapter 9 and the first five columns were taken as the principal components, giving $1024 \times 5$ for each filter response. This was reshaped into a $5120 \times 1$ feature vector, and the resultant 5120 data-points from the correlation phase were then used to train the Neural Network in the same fashion as in the preceding section. The input vector to the Neural Network thus became $5120 \times 563$ while the output layer remained a $4 \times 563$ matrix. The increased number of input neurons necessitated increasing the size of each hidden layer respectively to 600 and 200 neurons. The ANN structure is shown in figure 11.7. Initially, the magnitude value of the complex Gabor filter response was used, and second the phase angle between real and complex values was analysed.

Running the Gabor simulation on magnitude yielded the results seen in figure 11.8. Convergence took 542 epochs, and the final result was an accuracy of 58.3%. The process was repeated using the phase angle, giving an accuracy of 48.8% after 64 epochs. The lower number of epochs indicated that the software had to work much harder to find correlations in magnitude, and hence took much longer to converge. Despite the lower accuracy, the system trained much quicker using phase angle, indicating that phase data was easier for the system to find learning patterns in.
11.2 Principal Components Analysis.

11.2.2 Analysis of the Log-Gabor Filter.

The above process was repeated, using the Log-Gabor filter wavelets from figure 8.15 on page 103. The resulting PCA Magnitude Confusion Matrix is shown in figure 11.9. Convergence took 538 epochs. Accuracy using this method was 48.8%. The process was again completed using the phase angle. Results were 32.1% after 76 epochs. Again, magnitude yielded the higher absolute accuracy,
11.2 Principal Components Analysis.

but took much longer to find the correlations than it did for phase.

Figure 11.9: The Log-Gabor system confusion matrix, using the magnitude response. Convergence was after 538 epochs. It is interesting to note that most of the test errors indicate the software has mistaken the diseased sets as being healthy parts of the leaf (Output class 4).
11.2 Principal Components Analysis.

11.2.3 PCA Discussion.

From the results, both the Gabor filter and the Log-Gabor approach gave poor results both in phase and magnitude. This indicates that simply taking a section of the principal components of the raw filter data may not provide a good representation of the complete dataset. Despite the poor results, it was evident however that using the phase angle converged much faster than the magnitude value. This indicates that the ANN had to ‘work harder’ to find correlations in the magnitude data.

The following section analyses the Gray Level Co-occurrence Matrix data in a similar manner.
11.3 Analysis of the Gray Level Co-occurrence Matrix.

In accordance with section 9.3, each of the 40 Gabor filter responses was analysed as a separate GLCM, with both magnitude and phase correspondences once again treated separately. Visual analysis of the raw Gabor data\(^6\) indicated two GLCM offsets would best capture the data. Respectively these offsets were at 2 and 4 pixel spacings. GLCM angles were at 0°, 45°, 90°, 135°. Giving 8 GLCM responses for each of the 40 filter wavelets. Each respective GLCM response was then analysed for its Contrast, Correlation, Energy and Homogeneity. This gave 40 × 8 × 4 = 1280 data-points per leaf spot. In total this gave a 1280 × 563 input matrix to the ANN.

The systems were again analysed using a two hidden layer ANN with 600 and 200 neurons respectively. Results using the Log-Gabor filter offered a marginal improvement over the straight Gabor filter. The Log-Gabor results were:

- 61.9% accuracy with the magnitude response, after 20 epochs. See figure 11.10.
- 64.3% accuracy with the phase response, after 51 epochs. See figure 11.11.

\(^6\)As in figure 8.9 on page 96.
11.3 Analysis of the Gray Level Co-occurrence Matrix.

Figure 11.10: The GLCM magnitude confusion matrix. Convergence was after 20 epochs. Magnitude is still indicating a large number of disease spots are being classified as healthy.
11.3 Analysis of the Gray Level Co-occurrence Matrix.

![Figure 11.11: The GLCM phase confusion matrix. Convergence was after 51 epochs. The largest test error using phase is now between Rust and Tan spot. This is interesting as it indicates magnitude often confuses infected and healthy, while phase gives a better recognition rate that the spots are diseased, but is confusing which infection is present.](image-url)
11.3 Analysis of the Gray Level Co-occurrence Matrix

11.3.1 GLCM Discussion.

The GLCM method offered a marginal improvement over PCA in both phase and magnitude responses. This time however, phase took longer to converge, but gave a higher accuracy, which is in contrast to the results from PCA. This indicates that using texture features gives better data correlation for phase over simply taking the 5 principal components. The Log-Gabor filter also performed slightly better with the GLCM approach over the straight Gabor filter. Across all experiments, the use of magnitude in testing had a high rate of misdiagnosing the disease spots as being healthy. Using phase resulted in a better detection rate, but with many of the Tan spot samples being diagnosed as Rust. This indicates that the software identifies Rust and Tan spot as having a similar texture.

Despite an accuracy of 64%, it is evident that using texture data alone isn’t a good classifier of the disease spots. Adding the colour and spatial data back to provide colour + texture as a classifier is analysed in the next section.
11.4 Colour & Texture Composite Analysis.

In the preceding sections, it was shown that the initial colour analysis baseline gave the highest results, with PCA performing badly, and GLCM offering a marginal improvement over PCA for pure texture analysis. In an effort to improve the results, both the $1024 \times 1$ colour feature vector and $1280 \times 1$ GLCM texture feature vectors for each disease set were combined, producing $2304 \times 1$ features per disease spot. This process was continued for each image in the training set, producing an ANN input matrix of $2304 \times 563$. The ANN simulation was run, producing an accuracy of 97.6% after 41 epochs. The Structure of this ANN and the results are shown respectively in figures 11.12 and 11.13.

Figure 11.12: Artificial Neural Network training structure with two hidden layers and sigmoid triggering

---

Log-Gabor, phase, and GLCM.
11.4 Colour & Texture Composite Analysis.

Figure 11.13: The ANN result of using both colour and texture (GLCM, Log-Gabor, Phase) responses together. The test results were much higher than either colour or texture results on their own. The high correlation between training and validation accuracies indicates the system is correctly fitted.
11.5 Neural Network Discussion.

The preceding results indicated a rapid convergence and accurate results using both the colour and texture data, with the accuracy of both being greater than the individual components at $\approx 98\%$. The similar values for both testing and validation accuracy indicate that the system is correctly fitted, with two hidden layers consisting of 600 and 200 neurons. A complete table of the above analysis is presented in table 11.1.

<table>
<thead>
<tr>
<th>Type: Complex:</th>
<th>Hidden Network Neurons:</th>
<th>Epochs:</th>
<th>Training:</th>
<th>Validation:</th>
<th>Test:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Colour (Hue Channel)</td>
<td>-</td>
<td>200 - 100</td>
<td>114</td>
<td>97.5%</td>
<td>95.2%</td>
</tr>
<tr>
<td>Gabor PCA Magnitude</td>
<td>600 - 200</td>
<td>542</td>
<td>56.2%</td>
<td>42.9%</td>
<td>58.3%</td>
</tr>
<tr>
<td>Gabor PCA Phase</td>
<td>600 - 200</td>
<td>64</td>
<td>43.5%</td>
<td>46.4%</td>
<td>48.8%</td>
</tr>
<tr>
<td>Log-Gabor PCA Magnitude</td>
<td>600 - 200</td>
<td>538</td>
<td>62.8%</td>
<td>42.9%</td>
<td>48.8%</td>
</tr>
<tr>
<td>Log-Gabor PCA Phase</td>
<td>600 - 200</td>
<td>76</td>
<td>78.0%</td>
<td>38.1%</td>
<td>32.1%</td>
</tr>
<tr>
<td>Log-Gabor GLCM Magnitude</td>
<td>600 - 200</td>
<td>20</td>
<td>71.1%</td>
<td>71.4%</td>
<td>61.9%</td>
</tr>
<tr>
<td>Log-Gabor GLCM Phase</td>
<td>600 - 200</td>
<td>51</td>
<td>93.2%</td>
<td>72.6%</td>
<td>64.3%</td>
</tr>
<tr>
<td>Colour + Log Gabor GLCM Phase</td>
<td>600 - 200</td>
<td>41</td>
<td>97.0%</td>
<td>94.0%</td>
<td>97.6%</td>
</tr>
</tbody>
</table>
The conclusions from analysing the stock data images indicates that machine vision can quickly and accurately discern between the diseases present in wheat, when presented with both colour and texture analysis data. This result is perhaps intuitive, as the human eye and brain often requires both the colour of the image, and the texture to identify which disease is present. Further validation on ‘live’ images will follow in the next section.
11.6 Live Image Analysis.

The preceding section established that the use of colour and texture together provided an accurate diagnosis method on ‘static’ disease samples supplied by Dr Neate. This section focuses on actual photographs, and also validates both the ‘low cost’ approach of using a low-quality embedded camera with colour-compositing, as well as the ‘high-end’ approach of using High Dynamic Range Imaging on a professional SLR camera, which may be used by agronomists.

Due to the unavailability of actual diseased wheat samples, experiments in this section were carried out on three other monocotyledon plants, each showing signs of damage. While the ideal would be to use wheat samples, this section does however prove the robustness of the software: by using a data bank of ‘any’ disease, it is possible to extract the damaged spots and obtain a diagnosis, even across multiple plant species. This would prove beneficial for diseases such as Mildew, which can infect a wide variety of plants.

The samples used in this section are:

1. Nepenthes, showing strong signs of heat damage.
2. Agave, showing hail damage.
3. Clivia, showing insect damage.

The unedited in-situ photographs for each of these samples are shown respectively in figures 11.14, 11.15 and 11.16.
11.6 Live Image Analysis.

Figure 11.14: Nepenthes Leaves, showing heat stress.

Figure 11.15: Agave Leaves, showing hail damage.
11.6 Live Image Analysis.

Figures 11.17 through show 11.22 the results of compositing and extracting each leaf manually, using the grey card method. At the top of each pair is the colour-composite image, and below this the HDR image of the same leaf. Slight variations in alignment are due to the camera being hand-held in each case. Even before analysis it is interesting to note the side by side comparison between the lower-quality composite image, and the tone-mapped HDR. With the Nepenthes, the photograph shows an older leaf; while the composite image shows a slight yellow tint, the HDR has captured the full gamut of colour energy in this image and tone-mapping has enhanced it. This enhancement of unhealthy but not infected regions may be of particular benefit to agronomists, as disease-resistant strains often show discolouration in the regions where the plant’s immune system is fighting the infection.

The red squares denote the damaged regions identified by the k-means algorithm.

---

8The colour-composite images in this chapter were captured with the Samsung Galaxy Tab-3, using the internal 3MP camera. HDR images were captured with the 14MP Canon 5D SLR.

9See both section 5.3.2 and Appendix F for information on HDR enhancement methods.
11.6 Live Image Analysis.

Figure 11.17: Nepenthes Colour spot extraction. Image was constructed from three images, shot at 2,800K, 5200K, and 10,000K respectively.

Figure 11.18: Nepenthes HDRI. Image was constructed from three images, one at 0EV, and two ±2EV bracketed exposures.
11.6 Live Image Analysis.

Figure 11.19: Agave Colour spot extraction. Image was constructed from three images, shot at 2,800K, 5200K, and 10,000K respectively.

Figure 11.20: Agave HDRI. Image was constructed from three images, one at 0EV, and two ±2EV bracketed exposures.
11.6 Live Image Analysis.

The red borders around the damaged spots show good correlation between the two photographic techniques, with the Nepenthes and Clivia in particular showing a strong number of the same spots detected using both camera methods. However, a number of the smaller spots in the Agave were missed using both methods. This is most likely due to the spots not being badly damaged enough from the hail to show a strong difference in colour in the CIELAB ‘A-B’ channels: the leaf itself is still green with lighter regions which would become part of the ‘L’ channel and hence missed by K-means.

In the Nepenthes and Agave samples for both camera methods, there are some very large non-disease spots detected. On investigation this is due to a slight browning of the edges of the leaf also being detected by K-means. This is best shown in figure 11.23, which shows just the extracted regions from figure 11.18. As this boundary region is continuous the software detects it as a single very large spot; deleting any spots which cross the leaf boundary solves this problem. This filtering was done in software before saving the datasets for these leaves, ensuring only the leaf surface damage regions were presented to the log-Gabor filters.
11.6 Live Image Analysis.

Figure 11.23: Nepenthes total extracted region. The slight browning around the periphery and spine of the leaf have also been mistakenly extracted as spots due to the identical colouring. This can be avoided by deleting any segmented regions which overlap the leaf boundary.

11.6.1 Live Image Results.

The spots from each leaf were converted into a dataset for each camera, using both colour and texture (Log-Gabor + GLCM phase analysis) as defined earlier in this chapter. Once again the neural network structure consisted of two hidden layers, with 600 and 200 neurons per hidden layer, respectively. As such, both camera datasets were subjected to an identical classification system as outlined in section 11.4. Results are as follows.

Colour-Composite Images.

From the red borders in the above Colour-Composite images, the K-means algorithm extracted:

1. 92 spots from the Nepenthes
2. 69 spots from the Agave
3. 16 spots from the Clivia.
4. 257 healthy spots across all three leaves.

Giving a total of 434 spots across all four image sets.

The trained neural network for the colour-composite images is shown in 11.24. Final accuracy on the test confusion results was $\approx 94\%$.

![Colour Composite Neural Network](image-url)
11.6 Live Image Analysis.

**High Dynamic Range Images.**

A breakdown of the number of spots extracted from the HDR leaves is:

1. 66 spots from the Nepenthes
2. 31 spots from the Agave
3. 14 spots from the Clivia.
4. 444 healthy spots across all three leaves.

Giving a total of 555 spots across all four image sets.

It is interesting to compare these numbers to the number of spots extracted by colour-compositing; by comparison with the photographs (particularly the Agave; figures 11.19 and 11.20) the number of regions identified by the HDR algorithm indicated a stronger detection rate. Note however that the number of sample spots extracted (particularly from the HDR Agave) is much less than the red regions indicated in the photographs. This can be explained by the fact that most of these spots were small; any spots which overlapped the boundary, or contained only 16 pixels (denoting a $4 \times 4$ detection window) were discarded.

The final trained HDR Neural Network is shown in figure 11.25. Final training accuracy was $\approx 99\%$, with one Agave spot being misdiagnosed as healthy.
11.6 Live Image Analysis.

Figure 11.25: High Dynamic Range Neural Network. Final test accuracy is $\approx 99\%$, with one Agave spot misdiagnosed as healthy.
11.7 Live Samples Discussion.

Both the colour-composite and HDR imaging techniques yielded good results, with slightly higher accuracy obtained by the use of the better camera and High Dynamic Range/tone-mapping technique. It should be noted however, that despite the high results obtained from both methods, that in order to build a viable data-bank many more samples will be needed: while the network trained with as low as 14 Clivia samples in the HDR bank, in order to consistently detect this sort of insect damage, many more samples would be needed to train the neural network, so as to ensure that ‘any’ damage can be correctly recognised regardless of age or size of the spot. This will be discussed further in chapter 13.

The results from these live images indicate that HDR techniques with a high quality camera would be suitable for agronomists, and using camera augmentation techniques on low quality ‘smart devices’ is appropriate for growers.

The final design flowchart, including both camera alternatives is shown in figure 11.26. While the final system design protocol established in the next chapter will include an automated leaf extraction stage, the flowchart still shows this as dotted, indicating that it is a topic of future research as discussed in chapter 13.9.
11.7 Live Samples Discussion.

Image Capture Methods

- Colour compositing with Compact camera
- HDR Compositing with SLR camera

ICM

Initial leaf segmentation

- Depth from Stereo
- Manual Segmentation

Segmentation

Spot Subsegmentation

- CIELAB + K-means

Gabor Filter

Log-Gabor

Data Correlation

GLCM

Classification

Artificial Neural Network

Figure 11.26: The flowchart shows the final methodology of the complete system. Depth from Stereo is still to be finalised, as a topic of future research.
Chapter 12

Design and Implementation.

The preceding chapters focussed on establishing a ‘modular’ approach and independently analysing each aspect to identify the most efficient work-flow process. Chapter 11 analysed the complete system to establish the best performance. This chapter expands on this analysis and focuses on defining the final design protocol. This chapter will be presented in three parts: The first section will focus on using standard system development methodologies to define the final design protocol. The subsequent sections each focus on specific platform implementation, respectively using Matlab and Android.

12.1 System Development.

The end of chapter 11 defined the methodology of the complete system, as a result of analysing each stage and selecting the best approach from the preceding chapters. With a clear structure established, this section uses the sequential ‘waterfall’ method of system development.
12.1 System Development.

12.1.1 Requirements.

The core requirement is to develop an automated system to help growers and experts with disease recognition. Although elements of the core design are common to both growers and experts, both have their specific needs.

Grower Requirements.

The grower’s needs for the system are that it is intuitive to use, and completely automated so as to minimise training requirements, which the grower may not have time for. The final diagnosis must be unambiguous and easy to read, to minimise the risks of misinterpretation.

In the broadest sense, this means a completely automatic system. One envisaged scenario for use is that the grower walks into the field and notices the crop is showing signs of infection. The grower uses the app on the smart-phone to take an in-situ photograph of the infected region. Within a few seconds, the software displays the result, along with a probability of detection rate and information on disease treatment and other further steps to take.

Expert’s Requirements.

The requirements for use by agronomists are largely similar to that of the grower, however with access to a higher quality SLR camera connected to a smart device or PC, the extra enhancements that come with better equipment allow for greater scope of control of the information presented. The agronomist may already know the required mitigation steps, and so is more interested in other information about the disease. Information of use to the agronomist may be the software’s confidence (as a percentage) that the diagnosis is accurate, a measure of size and shape of the disease spots, how much of the plant is already infected: counting the total number of spots on the plant may help the agronomist estimate the rate of infection, while the actual position of the spots on the leaf may indicate
the disease strain. The HDRI enhancements of the infection may also assist the expert in deducing the risk of the infection spreading, as a plant showing strong signs of fighting the infection may require less treatment than a serious outbreak. The experts thus require a more comprehensive software suite, with options which aren’t required by the grower.

12.1.2 Design.

The system work-flow is presented in detail at the end of chapter 11. The final system design is:

1. The user captures images of the infection with an appropriate imaging device and method (over-sampling, stereo image sets).
2. The software automatically extracts a single leaf from the image, using stereo-from-motion techniques which require multiple image captures of the infected regions.
3. Individual spots are extracted by CIELAB and K-means Clustering.
4. The spots are analysed by Log-Gabor filters.
5. The Log-Gabor phase angle responses are correlated with GLCM.
6. The GLCM correlations are analysed by an Artificial Neural Network.

One aspect important to the design is the Graphical User Interface, i.e., how the complete program appears to the user, and how the user interacts with the software. It was mentioned above that the software should be intuitive and unambiguous to use. These two requirements mean that the ‘core’ system operation should be mostly hidden from the user, with only concise results available to the grower, and a more expanded set available to the expert.
12.1 System Development.

The Graphical User Interface.

Due to the automated nature of the software ‘workings’, the main GUI requirements are for the initial image capture stage, and the final results stage.

At the image capture stage, the software needs to photograph and extract an in-situ leaf from amongst the field of leaves. While this image capture stage is also automated, the GUI should at least show a ‘viewfinder’ screen to allow the leaf to be correctly centred in the camera frame. To help the software in this regard, it may be necessary to have an outline representation of a leaf, or some other aim-assisting marks on the screen to ensure the leaf of interest is correctly placed in the viewfinder. In order to correctly capture multiple exposures for over-sampling, it is necessary to have visual instructions on the screen as well: one to instruct the user to hold the camera still during each over-sampling (multiple capture) stage, and a cue to move the camera for stereo purposes. This cue may be connected to the camera motion sensors to ensure correct displacement for stereo capturing. A proposed Capture GUI is shown in figure 12.1.

![Camera viewscreen GUI](image_url)

Figure 12.1: The proposed Graphical User Interface for image capture. The center of the display shows an aiming point to assist in correctly framing the leaf in the camera shutter. The on-screen instructional cues indicate whether to hold steady during over-sampling captures, or to move between photos for stereo captures, possibly with an accelerometer feed to indicate the total distance to move. The green indicator illuminates when conditions are correct for capture, which may be automated once the indicator illuminates.
The second GUI of interest may just be a screen of text, with salient features such as the disease name, a visual description for quick validation purposes, and a treatment outline. This screen is shown in figure 12.2.

![Results GUI](image)

Figure 12.2: The proposed Graphical User Interface for displaying results. This will likely just be text, perhaps with a stock photo of the disease for visual validation. The GUI should also explain mitigation advice.

### 12.1.3 Implementation.

Platform specific implementation is covered in subsequent sections. During the experimentation stage, a ‘compartmentalised’ implementation in Matlab allowed for swapping of individual modules during the testing phase. Much of this software could already be reused with some refinement to the code, both in Matlab and in Android. Specifically, the Log-Gabor filters and Neural Network can be saved off as text (ascii-delimited, or comma separated value) files, and simply loaded as an array each time the software starts. This eliminates the need to recompute the Log-Gabor filter bank each time the software is run.

### 12.1.4 Verification.

Cropping production in Australia is a multi-billion dollar industry (Australian Bureau of Statistics 2006). As such, any automated system will need extensive trialling and close coordination with numerous growers and experts over several
seasons, in order to validate the system to the point where it can be a reliable analysis tool. There is an inherent risk involved with the grower relying on the software, with potential loss of crops if the advice given by the software is incorrect. Chapter 13 explains this point in further detail, and highlights the need for a larger data-bank of disease samples in the final system.

12.1.5 Maintenance.

By using a text file or similar soft-coded file for the Neural Network training structure and weights, it is proposed that the system will use a ‘live’ database, so that the infection database can be simply and easily updated by connecting to an on-line updating service and downloading the latest ANN training file. In this way, it is expected for the final system to remain up to date, and be able to recognise both a growing bank of diseases, as well as seasonal variations of existing infections.

By using a compartmentalised approach to the core software, it is also easier to refine or replace modules, as improvements in coding methods such as functionality enhancements, or hardware (camera) interfaces become available. Software updates, particularly automated updates on an Android device, would permit the latest version to be easily downloaded, ensuring both grower and expert have the most up to date version at all times.
12.2 Matlab Implementation.

Whilst Matlab is generally considered a mathematical tool, it has several useful optional ‘toolboxes’, for tasks such as Artificial Neural Networks. In keeping with the ‘modular’ approach, the software was designed to be as compartmentalised as possible. In Matlab this meant programming the software to generate the Log-Gabor filters, and then saving the filter bank itself as an ascii-delimited text file. This permitted the text file to be loaded when needed without recomputing the full bank each time. Stereo extraction was handled in another module, allowing this module to be swapped out with a working model at a later stage (see chapter 13 for a discussion on improving the stereo approach) CIELAB conversion and K-means extraction was also analysed in a separate module, allowing the disease spots to be extracted either using K-means, or another approach if required. Each of the disease spots was saved as its own $32 \times 32$ pixel file, permitting the later Gabor and Log-Gabor modules to be interchanged as required and to again work independently of the earlier software steps.

The final software module loaded the filter bank text file once, then loaded each separate $32 \times 32$ image. Both were passed to a function to filter with the Log-Gabor bank and perform GLCM on the phase response, before concatenating each spot’s filter response with the original colour data. This software module also performed ANN training. Appendix G shows the software pseudocode for the Matlab implementation.

12.3 Implementing on a Mobile Device.

The preceding chapters and methods were initially evaluated on a PC using Matlab. However for a portable system appropriate for growers to use in the field, it is necessary for the automated analysis to function on a smart device such as a mobile phone or tablet. This section discusses the platform-specific tools which will be required for implementing the protocol, while the companion
Appendix H contains the proposed Android pseudocode.

Having developed the software so far on the PC, it is possible to ‘reuse’ much of this method. Specifically, the Gabor filter banks which were mathematically derived in chapter 8 can be simply saved off as an array or comma-separated-value text file and soft-coded into the smart-phone software, eliminating the need for the phone to have to compute the filter banks each time the software is run. In a similar fashion, the Artificial Neural Network described in chapter 10 can be trained on the PC, and the trained weights incorporated directly into the software. If a future dataset is required (for instance, to include new diseases not currently found in Queensland, or to account for seasonal variations in the disease appearance), it will be necessary to replace the trained weights array, which can be implemented simply with a software update, or similarly by downloading a new trained ‘library’ with an expanded disease set.

Initial investigation into programming on the smart device has focussed on the Android platform, as this platform runs on a large variety of tablets and other smart devices. The development tools are freely available, using the Eclipse development package, and utilizing the OpenCV Machine vision package from within Android, to better access the phone’s camera functions. Other functions available on the phone may be leveraged for the final solution: While chapter 6 focussed on using the photographs themselves to deduce the motion between stereo images, on a mobile device it may be possible to use the on-board accelerometer and gyroscope sensors to measure this motion directly, aiding in stereo depth extraction. This concept will be mentioned again as a topic for future research, in chapter 13.9.

\[1\]The Eclipse Development environment was used for initial investigation into Android programming. (Eclipse Foundation 2014).

\[2\]OpenCV is a machine vision Application Programming Interface, from (Itseez 2014).
Chapter 13

Conclusions and Further Work.

A complete system has been designed which can recognize and extract separate infection spots on plant leaves, and can group those spots and compare to an internal database in order to diagnose and classify the infection/s present. The system designed mathematically emulates the function of the human eye, and uses artificial intelligence to respectively ‘see’ and ‘learn’ the infection characteristics in the same way a human analyst would. By integrating this system with a suitable camera, a complete detection system to automatically diagnose any diseases which the database is familiar with can be implemented.

This chapter consists of the following sections: sections 13.1 through 13.8 describe the conclusions of this research in accordance with the specifications defined in appendix A and chapter 1. An outline of further work to be completed and topics of future research is covered in section 13.9 of this chapter.
13.1 Review of Symptoms of Diseases.

Objective 1: Identify common leaf diseases and symptoms (e.g. Rust and Mildew) that occur in agricultural crops and specify at least three diseases to target for automated discrimination of a particular seasonal crop/s (e.g. Wheat).

Discussions with the agronomist Dr Steven Neate highlighted five main diseases which afflict Queensland wheat crops, with Stem Rust, Tan spot and Mildew selected for automated recognition. Chapter 2 covers an in-depth review of the prevalent diseases found in Queensland, and their current levels of severity. Whilst some of the diseases are currently considered low-threat, seasonal variations in both disease and crop strains has in the past lead to virulent infections wiping out 100% of the crops when the conditions have been right, and the infections have overwhelmed the plant’s immune system. Chapter 11.6 briefly discussed the use of advanced camera compositing methods to enhance damaged, but not visibly infection regions of crops: as crops such as wheat often show discolouration while the plant’s immune system is fighting the infection, this ability to enhance the discoloured regions is of benefit to both growers and professional advisers.

13.2 Identify Image Capture Methods.

Objective 2: Investigate different image capture technologies and techniques (e.g. mobile phone and digital single lens reflex cameras) for their ability to capture images of sufficient quality for leaf disease detection.

Image capture is the pivotal first step in image recognition. Research contained in chapter 5 and Appendix B indicated that the image capture quality was dependent both on the quality and type of the camera being used. The wide difference in results between using a high-cost Single Lens Reflex camera, and a low-quality integral camera embedded in a ‘smart’ device yielded two possibilities for application: using the low-cost approach to design a system that would work on a device (such as a phone) which a grower may already own, and the use of
more expensive camera equipment for professional analysts such as agronomists.

The differences between equipment necessitated experiments into just how the image quality (noise, and resolution) of the captured images affects disease detection rates. These experiments and also yielded several innovative approaches into improving the overall quality of the images by over-sampling the photographs. This over-sampling permitted respectively, an improvement in sensor quality on the embedded camera, and High Dynamic Range sampling techniques to improve the detection gamut, which can be used to highlight damaged but not visibly infected regions of crops.

13.3 Develop Segmentation Technique.

Objective 3: Develop and evaluate image segmentation techniques that achieve the following: (i) extraction of individual leaves or plants from the background, and (ii) separation of the extracted leaf/plant into healthy and diseased regions.

Image segmentation focussed on two sequential approaches. Initially a single leaf was extracted from the image where the background may have contained many other leaves. Once a single leaf was extracted, the infected regions on the leaf were sub-segmented using CIELAB colour space, which closely matched the function of the human eye, followed by k-means clustering analysis to identify the damaged portions of the leaf and extract them for further analysis.

Research into initial leaf segmentation focussed on biological cues: by taking photos from two locations a stereo, or binocular-vision approach was used to establish the depth of a single leaf, and thus use this depth map to mask out and discard the non-essential regions of the image. This research is contained in chapter 6 and Appendix C. Research into depth extraction is ongoing: while initial stereo rectification algorithms were able to give a rough depth extraction, it was found this approach needed further investigation to improve the results.

Whilst a fully automated method for extracting a single leaf using depth was
considered the ideal, overall the need to extract a single leaf could be achieved using alternative means. One such means which gave good results in the experimentation stage was to use a photographer's white balance reference card placed behind the single leaf, and to then use this card to mask out the leaf. This approach gave very accurate results during the experimentation stage. However in order for a completely autonomous diagnosis system to be designed, this method needs to be replaced with an automatic segmentation approach. These findings are expanded on in section 13.9.

Part (ii) of the segmentation objective called for investigation into sub-segmentation techniques to extract the individual infected regions from the surrounding leaf, and is presented in chapter 7. Two approaches were examined: initial research into colour segmentation was tried by simply making out any green regions. This process worked well in the test samples and was very rapid, however it was felt that a more robust approach was required, particularly in light of the fact that any discoloured (i.e., non-green) regions of the leaf would also be detected.

Research into alternative methods showed that by using CIELAB colour space and k-means clustering, that the infected regions could be accurately extracted, at the slight cost of increased processing time.

This chapter also continued the investigation into camera quality analysis, by artificially degrading the image resolution. It was found that both sub-segmentation approaches were able to detect the majority of spots accurately regardless of resolution, however at lower resolutions the boundaries between regions could be sufficiently blurred, so that two spots in close proximity may appear as one large conjoined spot.

13.4 Feature Detection and Classification.

Objective 4: Develop and evaluate techniques to filter disease symptoms (e.g. using pattern or template matching) and achieve correct classification (e.g. using...
13.5 Evaluation of the Proof-of-Concept System.

Feature detection and classification were treated as two individual topics. Investigation into feature detection again involved taking cues from human vision. It was found that the Gabor filter closely emulates the function of the human eye, with the related Log-Gabor filters offering an improvement in the low-frequency regions. Feature detection using Gabor filters is covered in chapter 8. The Gabor filter approach produces large amounts of data for each disease spot, and reducing the size of the data was found to be non-trivial. Efforts to correlate and find patterns in this data as a means of reducing the data size involved investigation into Principal Components Analysis and Grey-level Co-Occurrence matrices, with GLCM and Log-Gabor filtering ultimately giving the best detection rate.

The Log-Gabor and GLCM approach allowed the individual infection spots to give a consistent response to spots of the same type of disease. However in order to both group similar spots and to identify spots from a different infection, artificial intelligence in the form of Artificial Neural Networks was used to ‘teach’ the system to correctly identify the diseases. This research is covered in chapter 10 and expanded in the evaluation chapter 11. Results indicated that texture analysis alone was not a good indicator of disease type, and that some colour analysis data was also required. By using both colour and texture, detection rates upwards of 95% were obtainable.

13.5 Evaluation of the Proof-of-Concept System.

Objective 5: Develop a protocol for image-capture and automated analysis based on the developed machine vision system and evaluate the performance of the complete system.

The complete system was initially evaluated using disease samples supplied by Dr Neate. Detection and classification using both colour and texture yielded very
high levels of accuracy. Final performance was gauged on actual samples of plants. Due to the unavailability of actual wheat diseases, this section instead focussed on three other damaged leaves. The use of actual samples allowed both the evaluation of the ‘live’ image capture methods using over-sampling and initial leaf segmentation methods, as well as proving that the system itself is robust enough to work across a variety of diseases and even different plants. The implication of this is that by presenting the complete system with any new disease, the neural network can be retrained to recognise this new disease regardless of which crop or plant is currently being grown. This evaluation is covered in chapter chapter 11, with the design protocol established in chapter 12.

## 13.6 Spectral Analysis of Leaf Diseases.

*Objective 6: Determine if an improved disease detection can be achieved using spectral signatures of leaf diseases and spectral calibration of the image capture device*

Spectral analysis involved extracting each of the colour filter responses from the camera sensor, and over-sampling each of these separate coloured images using HDR to capture the full energy gamut of each colour. Research into this approach was initially encouraging, however the conclusion was that with only three spectral channels, that simply using the camera itself as a spectral sensor was a poor substitute for a genuine spectral camera, which may have upwards of 240 discrete spectral channels. One topic identified for further research however, may help in improving the spectral sensor capabilities of an SLR camera by converting it into a ‘Computed Tomographic Image Spectrometer’. Initial investigation into this equipment, as well as findings concerning colour filter extraction are explained in Appendix F.
13.7 Implement the Machine Vision System on a Mobile Device.

Objective 7: Implement the machine vision system on a mobile device and demonstrate the functionality of the developed system as a mobile device tool.

Investigation into mobile device implementation involved researching the Android software platform, and Eclipse integrated development environment. As the main focus of this dissertation was on establishing and validating the actual steps required to build a working system, implementation on the mobile device was considered secondary. Initial research into using the Android platform however focussed on what was possible using the mobile device, its camera, and the OpenCV machine vision package which can be used for many of the augmentation processes discussed above. This initial investigation is contained in chapter 12.3 and Appendix H.

13.8 Conclusions: Final System.

The final system design uses a work-flow process which closely matches the way the human eye and mind perceive the world. By using an appropriate image capture device and stereo vision techniques, a single leaf can be extracted from the scene. Further analysis of this leaf involves using a colour space which matches human vision perception of colour, filtering with a process that emulates the function of the optical striate cortex, and finally using artificial intelligence to correctly recognise the disease. After an initial training session where the software is presented with ‘known’ samples of each disease, it can be used to recognise any samples which may be presented to it in the field. By increasing the number of the diseases which the software is familiar with, the complete system will be able to recognise any disease or infection it is likely to encounter.
13.9 Further Work.

This section discusses both the outstanding requirements needed to ‘finish’ the system, as well as topics for future research.

13.9.1 Outstanding requirements:

There are a number of outstanding processes required before the system is ready for final implementation.

**Image Extraction:** Finding an effective method to automatically extract a single leaf from a field of leaves is desirable for the final system design. This may involve expanding on the research contained in chapter 6 to use depth, augmented with edge detection to provide better masking. On the mobile device it may be possible to leverage access to the on-board accelerometers and gyroscopic sensors, effectively allowing the camera itself to measure the displacement between image captures. This would simplify the depth from stereo process, and potentially provide another layer of accuracy: the on-board sensors may be able to augment or replace the creation of the fundamental matrix from the image itself, potentially reducing the ‘noise’ which was present in the depth experiments conducted in chapter 6.

Alternatively segmentation may involve a completely new approach such as deformable filters, or top down + bottom up segmentation as explained by (Kumar et al. 2010) and (Levin & Weiss 2009).

**Image Sample availability:** Another major point that needs to be addressed is the number of image samples. Throughout this research project, experiments were conducted on a limited number of samples, both of actual diseases, and of other damaged leaf samples. In order to develop a complete usable system, the number of disease samples needs to be increased. Ideally, building this large data-
bank of samples will include regional variations and crops with varying genetic
varieties, so that the final data-bank contains as many possible variations of the
disease’s physical characteristics as possible. One possible topic of future research
that may stem from this is that each year, agronomists need to analyse the genetic
resistance of the crops to ensure that seasonal mutations of the infections won’t
result in widespread crop loss. Often this is done by examining the size of the
’sickly’ regions around the pustules (if any) to discern how well the plant is
fighting the infection. By building up a large database of seasonal variation, this
tool may assist experts in their analyses of the mutations.

Mobile Device Implementation: This dissertation has focussed mainly on
investigating and validating a complete machine vision system. Ultimately, the
final system implementation is intended to be ported to the mobile device such
as a ‘smartphone’ or tablet. While initial investigation into these packages
was conducted, and pseudocode written\footnote{Refer chapter 12.3 and Appendix H.} actual platform implementation is
considered at this stage as a topic for further work.

Testing and Validation: Annual wheat production in Australia is a multi-
billion dollar industry (Australian Bureau of Statistics 2006). As such, any
automated disease recognition system would need to be extensively trialled
and validated, with the aide of numerous growers and agronomists before final
industry certification as a reliable analysis tool.

13.9.2 Future Research.

Building a Database: In the preceding sections it was indicated that a larger
number of samples would be needed to fully validate the software. A potential
topic for future research would involve making this sampling process ‘dynamic’
in the sense that if an online database was set up and maintained, then both the
variation and spread of crop diseases could be charted. The database would also
allow the software to maintain itself: any disease variations, or new instances of outbreaks could be quickly analysed and the disease data-bank on the grower’s devices updated with the new analysis. In this manner, any unknown disease outbreaks (such as imported foreign diseases) could be rapidly recognised and tracked, potentially allowing for eradication before having a chance to take hold.

Grading the plant’s immune system: While the research contained in this dissertation focussed mainly on recognising each disease regardless of the plant’s immunity, this tool could easily be expanded further into a classification system. Each year agronomists analyse the virulence of last season’s infections, and compare this to the genetic resistances of available wheat strains. In this manner part of the job of the expert is to suggest which strain should be grown in the upcoming seasons, to reduce the chance of infections destroying large regions of crop yield.

It is proposed, that with further development the automated software process could be used to ‘grade’ the sample’s resistance to infection, rather than just diagnosing the infections present. This may be possible by analysing the sickly regions around the infection itself, as plants with a strengthened immune system often show large sickly regions where the plant is fighting the subcutaneous infection, but may show little to no surface infection, as the fungus has been controlled before reaching the mature pustule stage. By working closely with experts to develop this tool over a number of seasons, it may be possible to heuristically plot future seasonal outbreaks, allowing preventative mitigation, or implementation of control protocols at the first sign of an outbreak.
Medical Applications: This project has focussed on agricultural applications. Whilst considered outside the scope of this current project, it is worth noting that the diagnosis software could potentially be adapted as a medical diagnosis tool, for example recognising glaucoma in the patient’s eye, and possible first stage diagnosis of skin cancers and other diseases. While this process would also need extensive acceptance trials, it is being considered as a topic for future research and system development.
References.


Bingham, A. & Mannila, H. (2005), Random projection in dimensionality reduction: Applications to image and text data, Laboratory of Computer and Information Science Helsinki University of Technology P.O. Box 5400, FIN02015 HUT, Finland.
REFERENCES.


Chojnacki, W., Brooks, M. J., Hengel, A. & Gawley, D. (2003), Revisiting Hartleys Normalized Eight-Point Algorithm., School of Computer Science, University of Adelaide, SA 5005, Australia.


Cook, J., McCool, C., Chandran, V. & Sridharan, S. (2006), Combined 2D / 3D Face Recognition using Log-Gabor Templates, Image and Video Research Laboratory Queensland University of Technology GPO Box 2434, Brisbane Qld 4000, Australia.


REFERENCES.


REFERENCES.


Appendix A

Project Specification.
FOR: Nathan Stern.

TOPIC: Machine Vision of Crop Diseases.

SUPERVISOR: Dr. Cheryl McCarthy.

PROJECT AIM: To develop a proof of concept proximal machine vision system that detects and discriminates leaf diseases in agricultural crops.


PROGRAMME:

1. Review of Symptoms of Diseases
   Identify common leaf diseases and symptoms (e.g. Rust and Mildew) that occur in agricultural crops and specify at least three diseases to target for automated discrimination of a particular seasonal crop/s (e.g. Wheat).

2. Identify Image Capture Methods
   Investigate different image capture technologies and techniques (e.g. mobile phone and digital single lens reflex cameras) for their ability to capture images of sufficient quality for leaf disease detection.

3. Develop Segmentation Technique
   Develop and evaluate image segmentation techniques that achieve the following: (i) extraction of individual leaves or plants from the background, and (ii) separation of the extracted leaf/plant into healthy and diseased regions.
4. **Feature Detection and Classification**  
Develop and evaluate techniques to filter disease symptoms (e.g., using pattern or template matching) and achieve correct classification (e.g., using artificial learning) for the specified diseases.

5. **Evaluation of the Proof-of-Concept System**  
Develop a protocol for image-capture and automated analysis based on the developed machine vision system and evaluate the performance of the complete system.

*As Time Permits:*

6. Determine if an improved disease detection can be achieved using spectral signatures of leaf diseases and spectral calibration of the image capture device.

7. Implement the machine vision system on a mobile device and demonstrate the functionality of the developed system as a mobile device tool.
Appendix B

Camera Operation.

This appendix covers the camera operation and file formats, and should be read in conjunction with chapter 5.

In order to understand the applicability of the various image capture methods and file formats available, a brief overview of the construction and operation of a basic camera is required. In all cameras, light enters the sensor through a lens, which directs the light onto a sensor. From Sensorcleaning.com (2010), the two prevalent sensor technologies are Charged Coupling Devices (CCD) and Complimentary Metal Oxide Semiconductor (CMOS) sensors: figures B.1 and B.2 show the basic architecture of the sensors. Fundamental differences between the two are that in a CMOS sensor, every pixel is read simultaneously by ancillary electronics and then the data transmitted to the output in parallel, while in the CCD sensor, the pixels are recorded and then transmitted one at a time directly from the sensor for processing.

At this stage it is sufficient to note that CCD sensors are more expensive, but produce images with lower noise, whilst CMOS sensors are cheaper to produce but can suffer from unwanted signal noise. CMOS sensors are traditionally used in compact cameras and embedded devices, while the higher-end SLR camera typically uses CCD technology.
Whilst the chipset architecture varies between CCD and CMOS, the basic function of the sensor remains the same in both cases: to convert light into an electrical signal. The construction of the sensor itself is explained in figure B.3. In brief, the lower grey layer represents the photo-diodes, with the layer above it representing the Colour Filter Array (CFA). The layer above this is an infra-red filter, which is used to reduce the natural sensitivity of the photocells to the IR light frequencies. The importance of both the IR filter and CFA layer are covered in appendix F.
Figure B.3: Basic construction of the optical sensor. The lower grey layer represents the photo-diodes. Above this are transparent filters, each sensitive to a particular light frequency. It is this repeating pattern of red, green and blue filters which allow the sensor to detect colour in an image. Above this is the infra-red filter. Often this is removed in machine vision applications, particularly in agricultural applications, where the IR-wavelength is useful for analysing vegetation stress. (Llewellyn 2014)

B.0.1 The Colour Filter Array (CFA)

Constructionally, all camera sensors are capable only of capturing monochrome images: each of the photo diodes discussed in the previous section records the absolute value of light that hit it at the time of capture.

Figure B.3 however indicated that by applying a pattern of filters to the sensor, each tuned to red, green, or blue light wavelengths, then each individual pixel of the sensor can be made responsive to that particular light colour frequency. The colour image can then be reconstructed from this composite response, using various methods.

The effect of this is best seen with an example as in figure B.4. The importance of the CFA is discussed in appendix F.
Figure B.4: Shows the process of capturing an image with a CFA. Image 1 represents the original scene. Image 2 shows the result of the sensor data; each node of the Bayer matrix has given an absolute response to the light wavelength it is sensitive to. Note the high values in the red channel in the flower, while the leaves have a higher red and green response. The data in Image 2 is what is saved in RAW format (12-bits). Image 3 shows how the processor colourises the image, based on the absolute response of each pixel and its associated colour filter. Image 4 shows the result of interpolating the missing pixels and saving as a JPEG image (8-bits). Camera used in this example was a Kodak EasyShare Z740 5MP.
B.0.2 Camera formats

After the camera sensor is exposed to light, the sensor surface builds up a charge relative to the amount of light that hits each pixel. This charge is then converted to a voltage, and the voltage for each sensor node read by the analogue to digital converter (ADC) before being saved off as a binary representation of the sensor data. The ADC in most cameras is 12 or 14 bit, while the sensor itself is usually slightly higher; some quantization occurs.

In cameras capable of shooting in RAW format\(^1\), this 12 or 14 bit data is saved directly to file, allowing the absolute value of each sensor node to be recorded. This results in lossless (Rowse 2014) uncompressed data. For photographic purposes this format often requires further editing on a PC to produce a visually acceptable image (e.g., adjusting the contrast, colour tone and sharpening) to best match the human perception of the image. However this post-processing necessarily alters the data from what the sensor originally captured\(^2\). For machine-vision purposes it makes more sense to utilise the unedited data. The merits of using RAW format are covered in Appendix F.

By comparison, JPEG format starts out with the same data as RAW format, however it is heavily processed within the camera to better match the image as the eye perceives it (see figure B.5) before being saved off as a much smaller 8-bit file: while some SLR cameras permit the saving of RAW + JPEG, most cameras either focus on just one or the other of the formats, or otherwise focus only on JPEG.

\(^1\)Almost all Digital SLR’s, and some high-end smart devices such as the Nokia Lumia 1520 are capable of RAW format capture. (Shankland 2013).

\(^2\)Rowse (2014) points out that RAW format is often admissible in court, while changeable image formats are not.
Figure B.5: Shows the relative sensitivity of the camera sensor and the human eye to the various light frequencies. In the natural state (i.e. RAW format), the camera is more sensitive to the higher wavelengths of light such as the Infra-red (IR) and near infra-red (NIR) bands. IR filters are usually fitted to the sensors to limit the IR-band sensitivity. In JPEG format, the camera data is reshaped so that the captured image more closely matches the colours that the human eye perceives. In machine vision applications, this reshaping is potentially undesirable. Image originally adapted from (maxmax.com 2013).

In a JPEG image, multiple steps are taken within the cameras processor (or pre-processor in mobile devices). The following list isn’t necessarily in order:

- From the Colour Filter Array response, the blank pixels (i.e., regions where there is no colour filter for that channel, as in figure B.4) are interpolated according to the nearest neighbour in a per-channel basis. The actual (recorded) pixels are often adjusted as well based on the newly interpolated pixels, and often the results are further modified by comparing the pixel response to the other 2 colour channels. The result is saved as the complete colour channel and the captured data from the sensor discarded. Although this interpolation procedure is based on the Discrete Cosine Transform principal, quite often the actual interpolation procedure is proprietary, and is closely guarded by the individual camera manufacturers. Unfortunately
this makes de-constructing back to the original CFA matrix impossible in most instances.

• Once every pixel on each of the colour channels has been saved, the camera usually processes the data further; white balance curves (discussed in chapter 5) are applied, and further colour channel shaping by comparison to the other colour layers can take place in order to produce a visually appealing image (i.e., colours in flowers are often enhanced if the camera is set to ‘vivid’ mode.)

• Some adjustment of exposure is usually performed, resulting in a ‘sharper’ image; shadows are often darkened slightly, or the blue channel in the shadows adjusted. This results in a reduction in the dynamic range of the image, and a higher contrast in the output.

• Image compression takes place; redundancy in the data is removed, as is anything that the human eye can’t perceive (such as variations in colour within shadow regions)

• As the final step in the JPEG process, each colour channel is compressed to 8-bits. This quantization stage further reduces the data that is available to each pixel in the image.

• All modification data is discarded, and the resulting image saved off as a 24-bit (3 x 8-bit) image.

• Any further adjustment or editing to the JPEG image within other software (including rotating the photo) results in a further loss of data.

From the above bullet points, it is evident that much of the original absolute data that is captured by the sensor is destroyed and cannot be recovered. It makes sense then that using a lossless format like RAW for machine vision techniques is a suitable approach. However, as noted above only the more expensive imaging devices can shoot RAW. Whilst the current market has high-end RAW-capable mobile devices, these were not available for research: the platforms outlined in the chapter 5 introduction were chosen because they represent the current standard.
Whilst the option of using high-end cameras is suitable for agronomists, it is expected that the grower’s will have access to JPEG-only devices. Chapter 5 is devoted to discussing techniques of improving the JPEG format on these devices, as well as improving sensor response.
Appendix C

Stereo Motion Analysis.

This appendix covers the mathematical derivations associated with building a 3-dimensional depth map from multiple images shot from different angles. Refer to chapter 6 for a discussion of the results of this research.

Stereo Analysis - The Correspondence Problem

When a 3-Dimensional scene is viewed from two disparate locations, the resulting images will show different views of the same scene. There are a number of geometric relationships between each point in the original 3D scene, and the corresponding point in the 2D images projected onto the camera sensor. This concept is best explained graphically.

Figure C.1 Shows the left and right images captured of a 3D scene consisting of two cuboid shapes.
Figure C.1: Left and right images of the same cuboid objects. Such images would be obtained in practice by either a pair of cameras mounted in stereo, or by one camera taking two shots from separate locations. (All CGI images by the Author)

The first step in the correspondence problem is to identify which points in the left image correspond to those in the right. This is often accomplished by using an edge finder algorithm, with a $9 \times 9$ pixel window function being used to identify the corners, and compare between images to locate the matching point (MathWorks, Inc. The 2010). These points are shown in figure C.2.

Figure C.2: Shows a birds-eye view of the 3D scene. The left and right images from figure C.1 are shown where the camera sensor plane would have been when the images were taken. Colour coded lines corresponding to each point in the left and right image are shown.
Once the corresponding points in both images are identified, it is possible to calculate the ‘epipolar’ geometry of the scene, as shown in figure C.3. By projecting the lines shown in figure C.2 through the camera plane, the focal point of each camera can be deduced. These are shown as the red nodes in figure C.3.

The epipolar planes are then taken as a plane running through these two distinct camera focal points, and also through the original correspondence point on the 3-dimensional geometry. Two are shown in the figure, with the rest hidden for clarity; it is evident that there are as many epipolar planes as there are geometric points visible between the two images.

It is easy to see that since all epipolar planes pass through the two camera focal points. This theoretical line connecting the two red nodes, called the ‘epipolar pencil’, is a line that lies on every one of the epipolar planes.

Figure C.3 also shows the camera vergence point, seen in grey in the left of the image. This is the theoretical point in 3-dimensional space that the cameras would be physically pivoted about; it is the intersection of the vectors taken normal to the image planes, and passing through the focal node.

Another important realisation of epipolar geometry is that any objects laying on the epipolar line between the correspondence point and focal point will appear
as a singularity to one camera, while the other camera will see the objects as distinct. This is shown in figure C.4 and is crucial to computing the 3D depth reconstruction.

Figure C.4: Shows a special correlation of epipolar geometry: any point laying on an epipolar line between the correspondence point, and the focal point of one camera, will appear as a single node to that camera. i.e. other objects on the same line will be ‘hidden behind one another’.

Having seen how the nodes in 3D space can be geometrically related to the points in the 2D planes, the question remains: is it possible given just the two images in figure C.1 to compute a set of image correspondences, and reconstruct the original 3D scene? The answer is yes; from Hartley & Zisserman (2000), given enough image correspondences \( x_i \Leftrightarrow x'_i \) between the two images, reconstruction is possible.

It is assumed that these correspondences relate to actual points \( X_i \) in 3D space, which are unknown. Similarly, the camera position, orientation and calibration of the cameras is not known \textit{a priori}. The task of 3D reconstruction is thus to find the camera matrices \( P, P' \) such that:

\[
x_i = PX_i \quad x'_i = P'X'_i
\]

for all points \( i \).

Depending on whether or not the cameras are calibrated, the above can be
achieved by computing either the essential or the fundamental matrix. Both are discussed below, however since stereo from motion relies on uncalibrated images, the essential matrix will only be reviewed briefly.

C.0.1 The Essential Matrix

In traditional stereo applications (Davies 2012), the two cameras are fixed with a known distance between them, so that the relationship between the two red nodes in figure C.3 is known, as is the vergence angle\(^1\) between the optical axes. In this set-up the cameras are said to be ‘calibrated’, and the transformation of corresponding points can be computed using the ‘essential matrix’ \(E = RC_x\) where:

\[
R = \begin{bmatrix}
R_\phi & 0 & 0 \\
0 & R_\theta & 0 \\
0 & 0 & R_\psi
\end{bmatrix}
\]

is the orthogonal matrix denoting the angles (in polar coordinates) between the cameras and the epipolar point.

\[
C_x = \begin{bmatrix}
0 & -C_z & C_y \\
C_z & 0 & -C_x \\
-C_z & C_x & 0
\end{bmatrix}
\]

is the translation matrix, denoting the Cartesian distances between cameras in their mounts.

However, while in a calibrated camera sense the above relationships are known \textit{a priori}, in the case of a single moving camera, the relationships need to be computed for each specific image pair.

\(^1\)Often in stereo images, the cameras are mounted parallel, thus there is no vergence between optical axes (Davies 2012)
C.0.2 The Fundamental Matrix

In the moving camera sense, the camera is considered as having six degrees of freedom (6-DoF). That is, it is free to translate in the \(X,Y,Z\) axes planes, as well as rotate in the \(\phi, \theta, \psi\) axes. Furthermore, in a complete sense any changes in camera zoom need to be taken into account, so that the complete \(Z-R-T\) transformation\(^2\) between images contains 7-DoF. It stands to reason then that in order to determine the fundamental matrix, a minimum of seven correspondence points across both images need to be defined. However, whilst the correspondence problem can be solved in some cases with as few as seven points using non-linear equations, the solution is often sensitive to the position of the correspondence points in the scene.

For this reason, a minimum of eight points is often chosen\(^3\). More than eight points is possible and generally leads to stability through redundancy. However this can lead to over-defined systems of equations, and extra steps are required such as taking a least-squares, or a RAmpdom SAmple COnsensus (RANSAC) approach.

The following derivation for the fundamental matrix is adapted from (Hartley & Zisserman 2000).

Given the relationship of correspondence points \(x_i \Leftrightarrow x'_i\), the points in each image can be expressed in matrix form as:

\[
x = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
\]

and

\(^2\)Zoom-Rotation-Translation.

\(^3\) Davies (2012) Considers the 8th degree of freedom as encompassing noise, or slight positional errors in the points themselves. It is assumed that these positional errors are independent zero-mean Gaussian about the ‘correct’ location in the image.
Then the fundamental matrix $F$ can be expressed as:

$$x'^T F x = 0 \quad \text{(C.1)}$$

Where $F$ is given as a $3 \times 3$ matrix, of rank 2.

Each point-match in $x_i \leftrightarrow x'_i$ results in one linear equation for the unknown $F$. Expressing $F$ by it's matrix components $f$:

$$F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

and expanding equation C.1 by multiplication:

$$x'x f_{11} + x'y f_{12} + x'f_{13} + y'x f_{21} + y'y f_{22} + y'f_{23} + x f_{31} + y f_{32} + f_{33} = 0 \quad \text{(C.2)}$$

factoring for the terms $f$:

$$(x', x'y, x', y'x, y'y, y', x, y, 1)f = 0 \quad \text{(C.3)}$$

where $f$ is the $9 \times 1$ column vector: $f = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{32} & f_{33} \end{bmatrix}^T$

Expressing eq C.3 in the general sense for any number of $n$ point matches:
Equation C.4 represents a homogeneous set of equations. For a solution to exist, \( A \) must have a rank of at most eight. If the rank of \( A \) is exactly eight then there exists a unique solution for finding the epipole. However in most instances there is no unique solution; due to noise or distortion in the camera lens or other factors resulting in ill-matched correspondence points, it is possible (given enough \( n \) points) for \( A \) to have a rank equal to nine. In this case, Hartley & Zisserman (2000) recommends using a least-squares approach to the solution.

The cause of this ill-matching can be blamed on a poor distribution of the homogeneous image coordinates. i.e. considering:

\[
x = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
\]

It is evident that since a modern camera sensor usually captures an image with dimensions ranging of a few thousand pixels\(^4\) a side, the first two terms in \( x \) will vary over a much larger range than the third in a typical image. The result of this is that if multiple points lay within a relatively small region of the image (say within 200 pixels), then the resultant vectors \( x \) of these points will be nearly coincident, and as a consequence \( A \) will be very nearly singular in that direction,

\(^4\)The Kodak Easyshare Z740 used in this report captures images consisting of 2,576 \( \times \) 1,932 pixels.
and small in the others.

Hartley & Zisserman (2000) proposed an adaptation to the basic 8-point algorithm, by normalising the images as a first step before computing the fundamental matrix. Hartley (1997) elaborates on the method. This normalisation term consists of two steps:

1. **Translation.** This is achieved by first finding the centroid of the image points, and setting the origin coordinates in the image to this point. This prevents small clusters of points from dominating the vectors.

2. **Scale.** This is achieved by scaling the image uniformly so that the mean distance from the new origin to any given point is \( \sqrt{2} \). This eliminates the ill-matching problem for the homogeneous image coordinates.

Formally the above steps can be considered as:

\[
\hat{x}_i = Tx_i, \quad \hat{x'}_i = T'x'_i
\]

where \( T, T' \) are the normalising transforms consisting of translation and scaling.

One final condition that needs to be met is that the \( 3 \times 3 \) matrix \( F \) is singular, and of rank 2. If \( F \) is not in fact singular, then the epipolar lines do not converge to a single point, and thus the camera focal point in the other image cannot accurately be determined. It is thus important to force \( F \) to be singular. This is achieved by finding the least-squares value of \( f \) as the singular eigenvector corresponding to the smallest singular eigenvalue of \( A \). That is, the last column \( V \) in the Singular Value Decomposition (SVD): \( A = UD \times V^T \).

Where, given \( A \) as a \( n \times 9 \) matrix:

- \( U \) is an \( n \times n \) matrix consisting of the eigenvectors of \( AA^T \).
- \( V \) is a \( 9 \times 9 \) matrix consisting of the eigenvectors of \( A^T A \).
- \( D \) is \( n \times 9 \) and the squares of the diagonal elements are the eigenvalues of \( AA^T \) and \( A^T A \).
The matrix $F$ is then replaced by the matrix $F'$. That is, the matrix that minimises the Frobenius norm $\|F - F'\|$ and with the condition $\det F' = 0$

To summarise these last few steps:

1. Normalise the data-points according to translation and scale.
   \[
   \hat{x}_i = T x_i, \quad \hat{x}'_i = T' x'_i
   \]
   Where $T, T'$ are respectively the normalising transforms consisting of translation and scaling.

2. Using $\hat{x}_i$ & $\hat{x}'_i$, construct $\hat{A}$ using equation C.4. Find the eigenvalues and corresponding eigenvectors of $\hat{A}$ using SVD. The eigenvector corresponding to the smallest eigenvalue is taken as solutions for $\hat{f}$.

3. Reshape $\hat{f}$ into the $3 \times 3$ matrix $\hat{F}$.

4. Perform a second SVD operation on matrix $\hat{F} = UD \times V^T$. This time take the two largest eigenvectors in $D$, and set the third (smallest) vector in $D = 0$ so that
   \[
   D' = \begin{bmatrix}
   \text{largest} & 0 & 0 \\
   0 & 2\text{nd largest} & 0 \\
   0 & 0 & 0
   \end{bmatrix}
   \]

5. Reconstruct $\hat{F}' = UD' \times V^T$. This $3 \times 3$ matrix $\hat{F}'$ satisfies the conditions of having rank 2 and $\det \hat{F}' = 0$

As a final step, the normalisation steps need to be removed so as to return the data-points back to the original (image) coordinate system:

\[
F = T^T \hat{F}' T
\]

Example:

As an example, consider again figure C.1. This time marking the distinct corners on the image as shown in figure C.5, and taking the coordinates of these corners, as shown in table C.1.
Figure C.5: Shows figure C.1, with the addition of numbered corners. These numbered corners correspond to their matching pair between left and right image.

Table C.1: Coordinates for the nodes shown in figure C.5 with $x, y$ representing the left figure, and $x', y'$ the right figure respectively. Note that in this instance, the origin point $(0, 0)$ is taken from the upper left corner of each image.

<table>
<thead>
<tr>
<th>Correspondence Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node.</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

From the coordinates given in table C.1, the centroid of the left image is $(144, 112)$ and for the right image $(189, 115)$. Subtracting these values from each node gives a new set of coordinates for left and right:
Table C.2: Coordinates from table C.1 with a translation of origin point.

<table>
<thead>
<tr>
<th>Centroid-Translated Correspondence Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node.</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

Next, the points are scaled so that the average distance from the origin is equal to $\sqrt{2}$:

$$ scale = \frac{\sqrt{2}}{\frac{1}{n} \sum_{i=1}^{n} \sqrt{x_i^2 + y_i^2}} $$

Which gives a value of 0.0133 for the left plane coordinates and 0.0137 for the right.
Table C.3: Coordinates from table C.2 with a change of scale, so that the average distance from the origin to each point is $\sqrt{2}$

<table>
<thead>
<tr>
<th>Node</th>
<th>$x$</th>
<th>$y$</th>
<th>$x'$</th>
<th>$y'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.4148</td>
<td>0.1494</td>
<td>-1.5810</td>
<td>0.1205</td>
</tr>
<tr>
<td>2</td>
<td>1.6819</td>
<td>0.0960</td>
<td>-0.1848</td>
<td>0.1752</td>
</tr>
<tr>
<td>3</td>
<td>0.3215</td>
<td>2.3235</td>
<td>-1.4715</td>
<td>2.3380</td>
</tr>
<tr>
<td>4</td>
<td>1.5352</td>
<td>2.1768</td>
<td>-0.1300</td>
<td>2.4749</td>
</tr>
<tr>
<td>5</td>
<td>-0.7990</td>
<td>-0.2107</td>
<td>-0.3080</td>
<td>-0.2765</td>
</tr>
<tr>
<td>6</td>
<td>0.2414</td>
<td>-0.2508</td>
<td>0.8008</td>
<td>-0.2354</td>
</tr>
<tr>
<td>7</td>
<td>-0.9457</td>
<td>-1.0244</td>
<td>-0.1574</td>
<td>-1.0978</td>
</tr>
<tr>
<td>8</td>
<td>0.0547</td>
<td>-1.0377</td>
<td>0.9240</td>
<td>-1.0978</td>
</tr>
<tr>
<td>9</td>
<td>-1.6793</td>
<td>-1.1044</td>
<td>0.6091</td>
<td>-1.2073</td>
</tr>
<tr>
<td>10</td>
<td>-0.8256</td>
<td>-1.1177</td>
<td>1.4989</td>
<td>-1.1936</td>
</tr>
</tbody>
</table>

From here, the corresponding coordinates can be loaded into A from equation C.4.

The resulting $10 \times 9$ matrix $A$ then used to generate matrices $U, D, V$. Where:
- $U$ is a $10 \times 10$ matrix, consisting of the eigenvectors of $AA^T$
- $V$ is a $9 \times 9$ matrix, consisting of the eigenvectors of $A^T A$
- $D$ is an $10 \times 9$ matrix, consisting of the eigenvalues of $AA^T$ and $A^T A$ on its diagonals.

Computing $D$ gives the smallest eigenvalue of 0.0057, and the corresponding
eigenvector:

\[
\hat{f} = \begin{bmatrix}
-0.0371 \\
0.0597 \\
0.0319 \\
-0.0164 \\
0.0262 \\
-0.6825 \\
0.0265 \\
0.7256 \\
0.0043
\end{bmatrix}
\]

Reshaping \( \hat{f} \) into the \( 3 \times 3 \) matrix \( \hat{F} \):

\[
\hat{F} = \begin{bmatrix}
0.0371 & -0.0597 & -0.0319 \\
0.0164 & -0.0262 & 0.6825 \\
-0.0265 & -0.7256 & -0.0043
\end{bmatrix}
\]

Again taking the SVD, where: \( U \) is a \( 3 \times 3 \) matrix, consisting of the eigenvectors of \( \hat{F} \hat{F}^T \):

\[
U = \begin{bmatrix}
0.0708 & -0.0607 & -0.9956 \\
0.2079 & 0.9771 & -0.0448 \\
0.9756 & -0.2038 & 0.0818
\end{bmatrix}
\]

\( V \) is a \( 3 \times 3 \) matrix, consisting of the eigenvectors of \( \hat{F}^T \hat{F} \):

\[
V = \begin{bmatrix}
-0.0272 & 0.0281 & -0.9992 \\
-0.9823 & 0.1847 & 0.0319 \\
0.1854 & 0.9824 & 0.0226
\end{bmatrix}
\]

and \( D \) is a \( 3 \times 3 \) matrix, consisting of the square roots of eigenvalues of \( \hat{F} \hat{F}^T \) and \( \hat{F}^T \hat{F} \) on its diagonals.
\[ D = \begin{bmatrix}
0.7305 & 0 & 0 \\
0 & 0.6817 & 0 \\
0 & 0 & 0.0399
\end{bmatrix} \]

Setting the last eigenvalue in \( D \) to zero and multiplying out the resultant equation gives \( \hat{F}' \)

\[
\hat{F}' = \begin{bmatrix}
0.0708 & -0.0607 & -0.9956 \\
0.2079 & 0.9771 & -0.0448 \\
0.9756 & -0.2038 & 0.0818
\end{bmatrix} \times \begin{bmatrix}
0.7305 & 0 & 0 \\
0 & 0.6817 & 0 \\
0 & 0 & 0.0399
\end{bmatrix} \times \begin{bmatrix}
-0.0272 & -0.9823 & 0.1854 \\
0.0281 & 0.1847 & 0.9824 \\
-0.9992 & 0.0319 & 0.0226
\end{bmatrix}
\]

Giving:

\[
\hat{F}' = \begin{bmatrix}
0.0026 & 0.0585 & 0.0310 \\
-0.0146 & 0.0261 & -0.6826 \\
0.0233 & 0.7257 & 0.0043
\end{bmatrix}
\]

Finally, denormalize the result to get back to the original image scale and coordinate system using:

\[
F = T'^T \hat{F}' T
\]

Where \( T \) and \( T' \) are of the form:

\[
\begin{bmatrix}
scale & 0 & scale \times \mathbf{-x_{centroid}} \\
0 & scale & scale \times \mathbf{-y_{centroid}} \\
0 & 0 & 1
\end{bmatrix}
\]

Thus:

\[
F = \begin{bmatrix}
0.0137 & 0 & 0 \\
0 & 0.0137 & 0 \\
-2.5803 & -1.5769 & 1
\end{bmatrix} \times \begin{bmatrix}
0.0026 & 0.0585 & 0.0310 \\
-0.0146 & 0.0261 & -0.6826 \\
0.0233 & 0.7257 & 0.0043
\end{bmatrix} \times \begin{bmatrix}
0.0133 & 0 & -1.9194 \\
0 & 0.0133 & -1.4912 \\
0 & 0 & 1
\end{bmatrix}
\]
Giving the total Fundamental Matrix $F$ as:

$$F = \begin{bmatrix}
0.0005 & 0.0107 & -0.8358 \\
-0.0027 & 0.0048 & -9.4941 \\
0.5287 & 7.1179 & 128.7735
\end{bmatrix} \times 10^{-3}$$

The fundamental matrix above can be used as a Z-R-T array, to compute the differences between each coordinate in the original images: By multiplying the images by this fundamental matrix, the images become reshaped (stretched, rotated and non-uniform scaled) until all corresponding points between left and right images lay along horizontal parallel points. Depth can then be extracted using trigonometric relationships. This application of the fundamental matrix to compute depth is covered in chapter 6.
Appendix D

Fundamentals of the Gabor Wavelet.

This appendix focuses on the derivation from first principals of the Gabor wavelet, initially in a 1-dimensional sense and then in 2D. Also included is the transformation into frequency (Fourier) space. Examples showing the multi-resolution properties of the filter are also included. This appendix should be read in conjunction with chapter 8.

From Chao (2010), the Fourier transform has to date been the most commonly used tool for frequency analysis. One downside of Fourier transformation is that the time-based information is lost and it is difficult to tell in dimensional space where a certain frequency occurs. In signal processing, this problem is commonly solved by using an appropriate ‘window function’. Gabor (1946) showed that among all kinds of window functions, the Gabor function proved to achieve the lower bound and performs the best analytical resolution in the joint domain.

Mathematically, the Gabor filter wavelet can be expressed as the function:

\[
g(x, y; \lambda, \theta, \varphi, \sigma, \psi) = e \left( -\frac{x^2 + \psi^2}{2\sigma^2} \right) e \left( i\left(2\pi \frac{X}{\lambda} + \varphi \right) \right)
\]  

(D.1)
Where:

$x, y$ represent the spatial coordinates of the filter.

$\lambda$ represents the wavelength of the sinusoidal factor.

$\theta$ represents the orientation of the normal to the parallel stripes of the Gabor function (rotation component).

$\varphi$ represents the phase offset in the argument of the cosine factor of the Gabor function. It is specified in degrees.

$\sigma$ is the sigma/standard deviation of the Gaussian envelope.

$\psi$ is the spatial aspect ratio, which specifies the ellipticity of the support for the Gabor function.

In order to understand equation (D.1) it is useful to consider the derivation in a 1-dimensional sense first.

At its base, the Gabor wavelet consists of a Gaussian curve, modulated with a complex sinusoidal wave. In standard form it is expressed as:

$$\varphi(t) = e^{-\alpha^2 t^2} e^{j2\pi f_0 t}$$  \hspace{1cm} (D.2)

Where:

$\alpha$ gives the sharpness of the Gaussian, and

$f_0$ is the modulated center frequency of the wavelet.

Graphically, the real component of this function can be seen in figure D.1:

\[1\] Since the Gabor filter is complex, this is actually a 2-dimensional sense. However the concept is easier to understand at this stage by only considering the real component of the filter in a 1D sense.
However, while the three distributions in figure D.1 have the same Gaussian area, they don’t meet the multi-resolution requirement: the size of the window should be dependent on the center frequency.

To achieve this requirement, we substitute $\alpha$ with $f_0/\gamma$, where $\gamma$ is a constant, and make the time duration of $\varphi(t)$ dependent on the central frequency $f_0$.

$$\varphi(t) = \frac{|f_0|}{y\sqrt{\pi}} e\left(-\frac{f_0^2}{\gamma^2} t^2\right) e^{j2\pi f_0 t}$$ (D.3)

The result of is shown in figure D.2:
Figure D.2: Example of $\varphi(t)$ with three different $f_0$ but the same $\gamma = 1$.

In order for this equation to be applied to an image, Equation (D.3) must exist as a 2-dimensional wavelet. Expanding (D.3) in 2D gives:

$$
\varphi(x, y) = \frac{f_0^2}{\pi \gamma \eta} \times e^{-\left(\frac{f_0^2}{\gamma^2}X_r^2 + \frac{f_0^2}{\eta^2}Y_r^2\right)} \times e^{(j2\pi f_0 X_r)}
$$

(D.4)

Where:

$$
X_r = x \cos (\theta) + y \sin (\theta), \quad Y_r = -x \sin (\theta) + y \cos (\theta)
$$

$X_r, Y_r$ give the polar value of each coordinate based on the orientation $\theta$ of the filter, while $x, y$ gives the horizontal and vertical coordinates of the 2D filter, with coordinate

---

2Actually a 4-Dimensional wavelet. However as there is no way to visualise a 4D figure, it will instead be convenient to view both real and complex components as separate 2D objects.
(0, 0) being central to the image.

$f_0$ is the center frequency.

$\gamma, \eta$ control the ellipticity of the Gaussian.

By setting the aspect ratio $\psi = \gamma/\eta$, and performing the substitution for bandwidth $b$:

$$b = \log_2 \frac{\sigma \pi + \sqrt{\ln^2 2}}{\sigma \pi \sqrt{\ln^2 2}}, \quad \frac{\sigma}{\lambda} = \frac{1}{\pi} \sqrt{\ln^2 2} \cdot \frac{2b^2 + 1}{2^b - 1}$$

eq (D.1) can be derived. However as the sigma/standard deviation term $\sigma$ cannot be specified directly and relies on the above transformation for bandwidth $b$, eq (D.4) was found to be easier to work with and hence was used for all subsequent calculations.

Graphically, eq (D.4) can be visualised in figure D.3:
Figure D.3: 2-Dimensional Gabor Wavelets. On the top is the real (cosine) component. On the bottom the imaginary (sine) component. Note the 90° phase offset between real and imaginary components: it is this phase quadrature that makes the Gabor filter so powerful.
D.1 Gabor Filter Response.

In order to understand the Gabor Process, this section will provide a brief overview of its usage, starting with the multi-orientation aspect of the filter.

Figure D.4 shows an input image, along with real and complex parts of a filter with $0^\circ$ orientation, and $0.2Hz$ frequency:

![Input Image with Real and Complex Parts](image)

Figure D.4: On the left, the input image. Center, the real component of the Gabor wavelet used for filter convolution. On the right, the imaginary component of the same wavelet.

The Input image is filtered\(^3\) by the Gabor wavelet using 2D filter convolution (Davies 2012):

\[
F(x, y) = f(x, y) * g(x, y) = \sum_i \sum_j f(i, j) g(x - i, y - j) \tag{D.5}
\]

\(^3\)Filter Convolution is described in chapter 8.3.1
The result of this filter convolution is shown in figure D.5

Figure D.5: The result of the filter convolution. On the left is the result of the real-component filter, and in the middle the imaginary-component filter response. Note that the banding straddles the input line in the real component, while in the imaginary component the center line is coincident with the original edge coordinates. The rightmost image shows the magnitude of both real and imaginary components.

From figure D.5 we can see that the Gabor filter gives a very strong response for image components which lay in the same orientation (rotation) as the filter, while anything outside the orientation is effectively removed from the output.

In order to detect the remaining edges of the octagon, it is necessary to use separate filters oriented appropriately, as in figure D.6
Figure D.6: The result of the filter convolution in 4 orientations. The upper four images show the filter and the individual convolutions. The lower image shows the total combined output.

From figure D.6, it can be seen that by running the filter multiple times with differing orientations, that the complete image response can be discerned.

The effects of altering the frequency will be explained in a further example: consider the input image shown in figure D.7. Test patterns similar to this one are commonly used for testing the optical resolution of imaging devices.
Figure D.7: Optical resolution test pattern. Created by quantizing the function $|\sin (\epsilon')|_{t=5.5}$.

Results of the convolution between figure D.7 and three discrete wavelets with varying frequency is shown in figure D.8.

Figure D.8: Result of convolving the filter with the test pattern, using the same $\theta = 0^\circ$ but varying the frequency. From left to right: frequency = 0.05hz, 0.2hz and 1hz.

Note from figure D.8 that at lower frequencies, the filter response becomes blurry, while at higher frequencies, other harmonic bands begin to appear.
D.2 Fourier Space Conversion.

In a 1-Dimensional sense, the Fourier transform of the Gabor equation can be represented by even and odd Gaussian curves in the complex frequency domain. By plotting both the real and imaginary components on separate axes, the result can be seen in figure D.9.

![Figure D.9: The complex-Fourier plot of equation D.3. The red line shows the real-symmetric component, while the blue shows the complex odd-symmetric.](image)

Using the linearity of the Fourier transform: \( \mathcal{F}\{\alpha f(A) + \beta f(B)\} = \alpha \mathcal{F}\{f(A)\} + \beta \mathcal{F}\{f(B)\}\), the complex component can be multiplied by \(i\) to bring it into real space, and then added to the real component. In this way, the even- and odd-symmetric components can be eliminated, resulting in a single real-space Gaussian curve with double the magnitude. This is represented in figure D.10. The Fourier transform of equation D.3 is thus simply expressed in equation D.6.

\[
\Phi(f) = \sqrt{\frac{\pi}{\alpha^2}} e^{-\frac{\pi^2}{\alpha^2}(f-f_0)^2}
\]  

(D.6)

---

\(^4\)see equation D.3 on page 223.
\(^5\)(James et al. 2011)
D.2 Fourier Space Conversion.

Figure D.10: The complex-Fourier plot of equation D.3 after converting both components to real space. The red line shows the real-symmetric component, while the blue shows the blue odd-symmetric. The black dotted line is the result of adding both real and complex components together, i.e. the Gabor filter exists only in one half of the frequency space. Note that the center of the image represents a frequency of zero. Moving outwards from the center represents an increase in frequency.

The same process can be followed in a 2-Dimensional sense: by adding the real and complex components in real space, the Fourier transform of equation D.4 can be expressed as in equation D.7

\[
\Phi(u, v) = e^{-\pi^2 \left( \frac{\gamma^2}{f^2} (ur - f)^2 + \frac{\eta^2}{f^2} vr \right)}
\]

\[
ur = u \cos(\theta) + v \sin(\theta), \quad vr = -u \sin(\theta) + v \cos(\theta)
\]

Where \( f \) is the frequency of the modulating sinusoidal plane wave and \( \theta \) is the orientation of the major axis of the Gaussian envelope, and \( \gamma, \eta \) control the ellipticity of the Gaussian. Figure D.11 shows both the 2-Dimensional Spatial and corresponding Frequency response.
Figure D.11: On the left is the real-space Gabor wavelet, with the corresponding Fourier transform on the right. \( f = 1.3, \theta = 0^\circ, \gamma, \eta = 1. \)

The relevance of using the Gabor equation in both the spatial and frequency domain to construct a full filter bank is covered in chapter 8.2.
Appendix E

Principal Components Analysis.

This Appendix expands on the Principal Components Analysis methods, discussed in chapter 9.

E.1 PCA - The Eigenvectors Approach

In order to explain the concept of principal components fully, it is necessary to consider a simplified example. Since it is impossible to visualise a 40th-dimensional dataset, the example will instead concentrate on a 2-dimensional dataset, which can be represented graphically on a Cartesian plot.

Consider as an example a random dataset\(^1\) This dataset represents two separate columns, which may have been taken as any real-world observations from two separate sources. It may be desirable to discern if there is a direct correlation between the two datasets: if both datasets increase together, if one decreases as the other increases, or if the two observations are completely disparate and hence there is no correlation.

\(^1\)This dataset and example are adapted from (Smith 2002).
As a first step to discerning any possible correlation, the data is plotted as $x$ vs. $y$.

See figure E.1.

From the plot, it is evident that as $x$ increases, so does $y$. The first step in computing PCA is to compute the average of each column and subtract it from each of the values: $(x_i - \bar{x}), (y_i - \bar{y})$. This produces a new dataset with zero
mean. This has the effect of moving the principal axes to the centroid of the data.

\[
\begin{array}{c|c}
   x & y \\
   0.69 & 0.49 \\
   -1.31 & -1.21 \\
   0.39 & 0.99 \\
   0.09 & 0.29 \\
\end{array}
\]

\[\text{Data} = \begin{pmatrix} 1.29 & 1.09 \\
                             0.49 & 0.79 \\
                             0.19 & -0.31 \\
                            -0.81 & -0.81 \\
                            -0.31 & -0.31 \\
                            -0.71 & -1.01 \end{pmatrix}\]

The next step is to compute the covariance of the data. Covariance is a measure of how much one column changes with respect to the other:

\[\text{cov} (x, y) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{n-1}\]

Since the dataset \(x, y\) is 2-dimensional, the covariance matrix will be \(2 \times 2\):

\[
\text{cov} = \begin{bmatrix}
\text{cov} (x, x) & \text{cov} (x, y) \\
\text{cov} (y, x) & \text{cov} (y, y) \\
\end{bmatrix}
\]

For the zero-mean dataset, this gives:

\[
\text{cov} = \begin{bmatrix}
0.6165 & 0.6154 \\
0.6154 & 0.7165 \\
\end{bmatrix}
\]

Finally, computing the eigenvalues and eigenvectors of the covariance matrix gives:

\[
\text{eigenvalues} = \begin{pmatrix} 1.28402 \\
                               0.04908 \end{pmatrix}
\]

\[
\text{eigenvectors} = \begin{pmatrix} 0.6778 & -0.7351 \\
                                   0.7351 & 0.6778 \end{pmatrix}
\]
By plotting the eigenvectors against the mean-zero dataset, we can make two interesting conclusions: firstly the principal eigenvector gives a ‘best line of fit’ through the dataset, and secondly, the eigenvector corresponding to the largest eigenvalue gives the principal axis, while the second eigenvector gives an orthogonal minor axis, as seen in figure E.2.

The significance of this is that the highest eigenvalue and its corresponding eigenvector gives the ‘Principal Component’ of the dataset. In other words, this line shows the axis along which the data most varies. By listing the eigenvalues in their descending order, the axes of variance can be also listed in descending order of relevance. Furthermore if a particular eigenvalue is small, it can generally mean its corresponding eigenvector can be discarded, reducing the dimensionality of the dataset. Whilst there is some data loss in this method, by retaining the most significant axes much of the data is preserved, resulting in a smaller dataset with only minimal compression loss.

Consider again the results from figure E.2. The minor axis in this instance may be deemed to be simply due to ‘noise’ in the original measurements. By discarding
the variance along the lower eigenvector the dataset can be reduced from 2-dimensional data down to just a single dimension with only minimal compression loss: all relative distances between points are still preserved.

In order to achieve this, first the axes of the original dataset need to be transformed (rotated) first from the original coordinates to the new orthogonal set. This is accomplished by simply multiplying the transpose of the eigenvector matrix, by the transpose of the zero-mean dataset. The results are shown in figure E.3

$$RotData = eigenvectors^T \times Data^T$$

![Figure E.3: Multiplication of the zero-mean dataset by the eigenvectors results in a change of axis for the data, so that now the major axis of the data lays on the rotated X axis.](image)

Similarly, if reducing the dataset is a requirement, it is only necessary to take the desired number of eigenvectors to be used in the rotation. Multiplying just the largest Principal Component eigenvector results in the dataset below, with the subsequent plot shown in figure E.4
Besides noise reduction, one other major use of PCA is in determining redundancy in signals. Redundancy can be thought of as multiple sensor readings of the same measurement: if redundancy in a system is high then there will also be a strong correlation between readings. It is this concept of redundancy which is most applicable to the task at hand: the Gabor wavelets filtered each pixel of the
input image 40 times. If there is a strong correlation between filter responses at each pixel then redundancy will be high and PCA will detect the ‘importance’ of the minor axes. In other words, it is this ability to discard the least valuable axes and hence reduce the dimensionality of the data which is of most relevance to the current task: by applying the above method to the 1024 × 40 Gabor-response feature vector, it is possible to minimise the ‘least valuable’ dimensions, potentially reducing\(^2\) the entire dataset down to as low as 1024 × 1.

### E.2 PCA - The Singular Value Decomposition Approach

In the preceding section, a method was derived for reducing the dimensionality of the dataset using eigenvalues and their corresponding eigenvectors. Whilst this method worked fine for small datasets, it was however anticipated that computing both the covariance matrix, and subsequent eigenvectors for a 40 × 40 matrix would be computationally expensive.

Shlens (2003) proposes an alternative method of developing PCA, using Singular Value Decomposition (SVD). The author notes that SVD presents a more general form of change of axis, and states that “SVD is so closely linked to PCA that often the terms are used interchangeably”. It was decided to trial this method in parallel with the eigenvectors approach to see if there was any improvement in the results.

Consider again from section E.1 the equation for covariance:

\[
\text{cov} (x, y) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{n-1} \tag{E.1}
\]

As a first step, this equation can be rewritten in terms of the matrix variables

\(^2\)Chapter 9.2.1 explains that taking the top 5 dimensions is the most practical solution, i.e., 1024 × 5.
of the zero-mean dataset, by treating all columns (measurements) and rows (separate sensor readings) as a single $n \times m$ matrix:

$$\text{cov}(\text{Data}) = \frac{1}{n-1} \text{Data} \times \text{Data}^T \quad (E.2)$$

Making the substitution $X$ for $\text{Data}$:

$$\text{cov}(X) = \frac{1}{n-1} XX^T \quad (E.3)$$

A change of data can be considered by defining $Y$ as:

$$YY^T = \text{cov}(X)$$

Then $Y$ can be derived as:

$$Y = \frac{1}{\sqrt{n-1}} X \quad (E.4)$$

By applying equation E.4 to the zero-mean dataset $\text{Data}$, the following table of values is computed:

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.230</td>
<td>0.163</td>
</tr>
<tr>
<td>-0.437</td>
<td>-0.403</td>
</tr>
<tr>
<td>0.130</td>
<td>0.330</td>
</tr>
<tr>
<td>0.030</td>
<td>0.097</td>
</tr>
</tbody>
</table>

$Y = \begin{bmatrix} 0.430 & 0.363 \\ 0.163 & 0.263 \\ 0.063 & -0.103 \\ -0.270 & -0.270 \\ -0.103 & -0.103 \\ -0.237 & -0.337 \end{bmatrix}$
The SVD can now be computed for $Y$.

\[(\text{SVD}): Y = UD \times V^T.\]

Where, given $Y$ as a $10 \times 2$ matrix:
- $U$ is an $10 \times 10$ matrix consisting of the eigenvectors of $YY^T$.
- $V$ is a $2 \times 2$ matrix consisting of the eigenvectors of $Y^T Y$.
- $D$ is $10 \times 2$ and the squares of the diagonal elements are the eigenvalues of $YY^T$ and $Y^T Y$.

Of the values returned from the SVD, the most valuable are $V$ and $D$. The values of $V$ are:

$$V = \begin{bmatrix} -0.6779 & 0.7352 \\ -0.7352 & -0.6779 \end{bmatrix}$$

While $D$ gives a $10 \times 2$ diagonal matrix, the non-zero elements on the main diagonal of which are:

$$D = \begin{pmatrix} 1.331 \\ 0.222 \end{pmatrix}$$

Note that these eigenvalues are the square roots of the eigenvalues computed in the preceding section. This is understandable given the $\sqrt{n-1}$ term in equation E.4.

As in the preceding section, finding the principal components of the dataset involves rotating (multiplying) by $D$. This gives:
\[
\begin{array}{l|l}
 x' & y' \\
-0.828 & 0.175 \\
1.777 & -0.143 \\
-0.992 & -0.384 \\
-0.274 & -0.130 \\
\end{array}
\]

Note that the left column gives the same results for Principal Components as in section E.1, albeit with an opposite sign due to a swapped sign on the rotation eigenvector.
Appendix F

Bayer Filter Extraction.

This appendix is presented to meet requirement 6 in Appendix A:

*Determine if an improved disease detection can be achieved using spectral signatures of leaf diseases and spectral calibration of the image capture device.*

The work in this appendix presents an entirely innovative approach to using an unmodified camera as a spectral sensor, by building on the process described in (Lebourgeois, Bgu, Labb, Mallavan, Prvot & Roux 2008). The method in the reference used Bayer Filter Extraction on RAW format files to produce three separate images, each containing just the spectral sensor response from the camera’s inbuilt Colour Filter Array. The research presented in this appendix expands on this method by performing Bayer Filter Extraction across a range of images, and using High Dynamic Range Imaging methods to capture the total energy gamut across each colour bandwidth.

In Appendix B the concepts of the colour filter array attached to the camera sensor were discussed, along with the differences between RAW file format and JPEG. Chapter 5 explained the use of High Dynamic Range Imaging for improving the detection of disease spots.

In this appendix, these concepts will be expanded further. By isolating and comparing the separate pixels from the colour filter it was theorised that it was
possible to analyse the spectral signature of the leaf disease, and thus improve on the disease detection.

This appendix concludes with an overview of research conducted into converting a commercial camera into a hyperspectral imaging device, given in (Habel, Kudenov & Wimmer 1985).

In Appendix B the distinction between RAW format and JPEG was expressed primarily in the way that JPEG interpolates and alters the data captured by each pixel on the camera sensor. The algorithms behind this interpolation are proprietary property of the camera manufacturers, and are usually established at the camera prototype stage by comparing scenes taken with the camera against a known image or colour reference. Conversely, RAW format simply saves off the absolute pixel responses to each node on the image sensor, without any processing. RAW format isn’t an image ‘per-se’, rather it is a direct recording of what the electronic sensor sampled. Conversion into a visible image requires software processing, which is usually performed manually by the photographer, and is analogous to chemically processing traditional negative film (Rowse 2014).

While an in-depth discussion of the colour composition of light is beyond the scope of this report, it is sufficient to give an example: in nature it is entirely possible to produce pure monochromatic coloured light (say, orange) while the camera sensor must use a combination of its red, green and blue pixels to record this; a JPEG representation of this colour may actually result in RGB values that are quite different to those captured in RAW, in order for the colour to appear to the human eye as being the same.

This can be explained graphically by comparing the spectral sensitivity response of both the human eye and the camera, seen in figures F.1 and F.2. A full discussion is provided in references (Koren 2014) and (maxmax.com 2013).
Three cone types (\(\phi\), \(\gamma\), \(\beta\)) correspond roughly to R, G, B.

Figure F.1: Human Spectral sensitivity to colour. (Image adapted from (Koren 2014)).

Figure F.2: Adjusted RGB curve, Spectral response of a Canon 40D camera. Image adapted from (maxmax.com 2013). From the reference, the spectral response was produced by exposing the sensor of the camera to a tunable monochromator filter, fitted with a polychromatic light source.

The proposed method of using a standard DSLR (or any camera capable of saving
RAW format) as a multi-spectral sensor requires a quick review of the Colour Filter Array (CFA). In Appendix B it was explained that the camera sensor itself is monochromatic, with a matrix of alternating colour filters applied over the top of the sensor in order to tune the individual pixels to a particular wavelength sensitivity. The concept of the CFA is shown in figure F.3.

![Figure F.3: The process of recording different channel data using the Colour Filter Array. Note that in a Bayer Matrix sensor that there are two green channels.](image)

It was noted in Appendix B that the JPEG format uses an interpolation process to fill in the missing (white) pixels shown in figure F.3. However, in RAW format, these individual sensor responses can be retained, and furthermore using Bayer Filter Extraction\(^1\), it is possible to divide the channels into separate images. From (Lebourgeois et al. 2008), extracting each colour channel into its own image results in four images (R,G,G,B), each half the size of the original photograph, but composed only of the raw pixel response to the wavelengths those pixels were sensitive to, without interpolation between layers. This results in a ‘pure’ sensor response image for each of the four channels, each containing only the spectral

\(^1\)Bayer Filter Extraction can be accomplished by direct matrix array addressing on the RAW image. For example copying every second pixel in Matlab to a new matrix, or performing a similar operation in Android.
bandwidth over which that particular sensor was tuned to.

It was mentioned in chapter 5 that the use of HDRI techniques allows the absolute quantity of light striking each pixel to be recorded. By combining HDRI with Bayer Filter Extraction, it was proposed that the absolute intensity of light of a particular wavelength striking each node on the sensor can be recorded as three (or four, if two green channels are required) separate HDR images. By comparing the tonal range across all three channels with the particular calibrated response curve for the camera given in figure F.2, it was hypothesised that the actual wavelength of light for each part of the image could be deduced.

This process was trialled with a canon 60D camera, with the Infra-red filter removed to allow sensitivity into the IR range. Results are shown in figures F.4 to F.7. The actual process for generating these images was:

1. The camera was initially set to take pictures in RAW format mode.

2. 5 images were taken of the scene, with an EV range of +/-3EV

3. Each separate image had the four colour images extracted from the Bayer filter (one green channel was discarded) The result was 5×3 images, differing in EV and colour respectively.

4. Each of the separate colour channels was then merged across the EV range into a HDR image, and finally tone-mapped for display on a monitor. The result was 3 x HDR images, each one showing the absolute sensitivity to light of a specific colour channel.
Figure F.4: The original colour image, containing all three colour channels.

Figure F.5: The HDR-merged Blue channel. The blue CFA response was extracted from each of the 5 RAW images, and HDR techniques were used to merge and tonemap. From figure F.2, The values in this image represent the spectral range between $\approx 370 - 500$ nm.
Figure F.6: The HDR-merged Red channel. The red CFA response was extracted and merged using HDR, giving the spectral sensitivity over the 550 – 950nm range.

Figure F.7: The HDR-merged Green channel, sensitive to the 450 – 650nm range.

The figures F.5 through F.7 represent the full colour bandwidth between 370nm and 950nm. It was proposed that there was sufficient overlap between the bandwidths across all three images, to measure with some accuracy, the actual
per-pixel light frequency. A sample was cropped from the above images, and is seen in figure F.8. Comparing the colour responses across all three channels indicates that this plant appears as a high-midtone colour in the red and green channels, (slightly higher in the red) and a low tone in the blue channel. Comparing this with Figure F.2 would suggest this region of the image is strongest in the 550-570nm band.

Figure F.8: A cropped sample from the middle region just left of center.
Whilst the method described gives a broad spectral value, it seemed unlikely that accurate spectral measurement could be discerned simply from this method, even with HDR augmentation: The estimated spectral range is broad enough that it could have simply been estimated from the colour photograph itself. This is to be expected: considering that the camera only possesses three channels, while a dedicated hyperspectral sensor may have upwards of 240 channels (Resonon 2014). According to maxmax.com (2013), the CFA is also rarely perfectly constructed, allowing some leakage between sensor nodes. This is graphically shown in figure F.9, where the sensor was exposed to monochromatic green light for shutter times varying from 1 to 30 seconds. Even when exposed to pure green light, the red and blue sensors detect some signal, particularly at longer shutter times\footnote{Whilst the source doesn’t indicate as such, this could be at least partially attributed to stray gain as the sensor heats up during long exposures. For this reason cameras intended for long exposure shots such as in astro-photography often use sensors fitted with cooling units (Hyperion Cameras 2012).}. The image also shows the non-linear response of the sensor, which would also need to be calibrated for on a per-device basis.
Figure F.9: The Green Power Curve spectral response from a Canon 40D camera. Note the ‘leakage’ of the other channels at higher exposure times when the sensor is exposed to pure green light. Note also the non-linear sensor response. Image adapted from (maxmax.com 2013).

Another possible factor as to why the sensor cannot be used as a pure spectroscope is because whilst in the lab a pure light source is available, in an actual environment the light bounced off a surface is a mixture of multiple light wavelengths. As such attempting to measure a single value is impractical.

One possible solution is to attempt to break up the white light into its spectral components. This can be achieved by fitting a Holographic Diffraction Grating (Edmund Optics 2014). Fitting a diffraction grating to the camera effectively turns it into a linescan device: if the grating is mounted horizontally, then the vertical axis of the sensor displays the spectral response from the scene while the horizontal axis remains a spatial dimension (RSpec 2014).

As a variation of the diffraction grating process, is ‘Computed Tomography Image Spectroscopy’ (CTIS). In CTIS, a 2-dimensional diffraction gel is used, allowing
the camera sensor to retain its spatial response in both axes, but also adding an extra axis for spectral response. The resultant dataset is thus a 3-dimensional response. Research into converting an SLR camera into a CTIS capable of hyperspectral imaging from a HDR capture has been conducted by the Vienna University of Technology, and the University of Arizona (Habel et al. 1985). Images of the camera and its basic operation are shown in figures F.10 and F.11.

Figure F.10: Prototype Hyperspectral CTIS, converted from a Canon 5D SLR. Image source: (Habel et al. 1985)
Figure F.11: Diffractions interpreted as parallel projections. The lower component of the figure shows the 2-dimensional pixels recorded by the camera sensor. The 3-dimensional Voxel cube is then reconstructed from the 2D image. Image source: (Habel et al. 1985)

Despite the interesting prospects, it was decided that converting an SLR camera into a functional spectrometer was outside the scope of this project specification. Initial research into the topic indicates that it may be a viable process to aide in disease detection, and thus it is suggested as a topic for further work in a future project.
Appendix G

Matlab Pseudocode.

Below is the pseudocode representation for the Matlab implementation of the program, as described in chapter 12. The software was programmed in four parts, as follows.

**Log Gabor filter:**

DEFINE variables for overall shaping functions: scale, orientation, number of wavelets.

DEFINE The size of the complete filter bank array.

DEFINE The polar coordinate space for each wavelet.

FOR number of orientations (8 total).

FOR number of frequencies (5 total).

    COMPUTE each Fourier log Gabor value in polar space, using frequency, orientation, and the shaping functions.

    SAVE each filter off into the appropriate place in the filter bank array.

SAVE the final filter bank off, as a comma-separated-variable text file, ascii-delimited text file, or similar.
**Stereo Extraction**

CAPTURE Images. 3 left, 3 right in exposure-bracketed groups.

LOAD the center (0EV) image from each set.

COMPUTE the fundamental matrix based on two images.

COMPUTE the depth map and use it to generate masks.

TRANSFORM the masks back to image space using the inverse fundamental matrix.

COMBINE each set of 3 images using HDR algorithms.

APPLY the masks to the HDR images.

IF Tonemap if needed for processor constraints.

ELSE Store the 32-bit float image.

SAVE the singly extracted (masked) HDR leaf image.

**K-means extraction**

LOAD the segmented HDR leaves.

CONVERT the colour space to CIELAB.

DEFINE The K-means criteria: number of clusters, distance criteria, cluster overrun handling, etc.

PERFORM the K-means cluster analysis on the CIELAB leaf.

DEFINE which cluster contains the spots.

FOR each spot on the leaf:

DEFINE a bounding box around the spot on a per-pixel basis

CROP the original RGB colour image to the bounding box.

IF the pixel count in each box is < 16 or overlaps the leaf boundary, discard this spot.

FOR the valid spots, reshape the cropped region to $32 \times 32$ pixels.

SAVE each spot as a separate image.

FOR the remaining healthy regions of the leaf, extract a number of $32 \times 32$ pixel healthy leaf samples.

SAVE the healthy samples.
**Gabor Filtering and Neural Network**

LOAD the Ascii-delimited text file containing the Log-Gabor filter bank.

FOR each disease set:

    FOR each disease spot:
        LOAD each of the $32 \times 32$ pixel spots. Convert to
        HSV colour space.
        CONVERT the hue channel into Fourier space.
        FOR each Gabor filter wavelet, multiply the Fourier spot.
        CONVERT the filtered spot back into the spatial
domain with the inverse Fourier transform.
        COMPUTE the 8 GLCM responses for each response.
        CONCATENATE each of the GLCM responses for each of the
        40 filtered images.
        RESHAPE the original colour image hue channel into a single
column vector, and concatenate with the GLCM response to produce
a single
colour + texture vector per spot.
        SAVE each column vector.
        COMBINE each of the column vectors for each disease to produce
an input matrix.

COMBINE each matrix for the disease set into one large input matrix, ensuring
the order of spots is preserved.

IF the matrix is being trained for the first time:

    CREATE a new matrix with the same number of columns as the input
matrix, this is the output matrix.
    DEFINE the Neural Network parameters: number of neurons in each layer,
    convergence criteria, etc.
    RUN the neural network on both the input and output matrices.
    SAVE the final neural network training weights.

ELSE if the disease sample is an unknown sample and the network is trained.
LOAD the Neural Network trained weights.
APPLY the input matrix to the trained Neural Network.
DISPLAY: the final disease result, along with the percentage of error confidence.
Appendix H

Mobile Device Pseudocode.

Below is the pseudocode for an Android implementation of the program, as described in chapter 12.

Camera Image Capture:

FOR Left/Right Image.

LOOP Multiple times according to desired number of images.

CAMERA captures photos, at different Exposure Value (EV) settings.

WRITE one image from each left/right set to storage.

MERGE photos using High Dynamic Range Image algorithms.

WRITE High Dynamic Range Image to device storage.

LOAD the two single left/right images.
Perform stereo vision computations on these single images to compute the fundamental matrix.
Use the fundamental matrix to compute the depth map.
Use the depth map to extract the single leaf from the HDR image (this method is more efficient than trying to compute depth directly from the 32-bit floating point HDR images).

WRITE The single Extracted HDR Leaf to device storage.
**Image Preprocessing:**

READ HDR leaf image from storage.
Convert HDR Composite image to CIELAB Colour space.
K-means cluster analysis Performed on the converted image.
Isolate the cluster $k$ which contains the diseased spots.
FOR Each spot in the cluster:
   - Compute the bounding box for each spot.
   - Use the bounding box to extract the spot.
   - Reshape each spot to $32 \times 32$ pixels.
   - WRITE each spot image to storage.

**Image Filtering:**

LOAD the Gabor Fourier Bank.
FOR Number of saved spot images $n$:
   - Convert each spot to Fourier Space.
   - FOR 1 to 40:
     - Array-address each Gabor wavelet from the Gabor Filter bank.
     - Multiply The Fourier spot by each Gabor Fourier Wavelet.
     - Perform an Inverse Fourier Transformation to convert the filter response back to the spatial domain.
     - Compute the phase angle for each spatial response.
     - Compute the GLCM for each phase response.
     - WRITE Each GLCM response, appending each to the end of the same array to construct a $1280 \times 1$ feature vector.
   - Convert each spatial spot to HSV colour space.
   - Reshape the Hue channel from the HSV into a $1024 \times 1$ feature vector.
   - Append this feature vector to the complete vector from the GLCM response, to create a $2304 \times 1$ vector for each disease spot.
WRITE a $2304 \times n$ matrix consisting of the feature vectors, where $n$ is the number of spots.
Artificial Neural Network:

LOAD the trained weights for the neural network, either from an internal array or a separate downloaded disease bank.
LOAD the spot data matrix for the leaf.
Run the spot data matrix through the Neural Network.
DISPLAY the results of the Network diagnosis on the phone screen.