FEATURE-BASED POSE-ESTIMATION FOR ROBOTIC FIXTURELESS ASSEMBLY

by

Christopher Stewart Langley

A thesis submitted in conformity with the requirements for the degree of Master of Applied Science, Graduate Department of Aerospace Science and Engineering, in the University of Toronto

© Copyright by Christopher Stewart Langley 2001
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-58731-2
Title: Feature-Based Pose-Estimation for Robotic Fixtureless Assembly
Degree: Master of Applied Science
Year Conferred: 2001
Name: Christopher Stewart Langley
Department: Aerospace Science and Engineering
University: University of Toronto

ABSTRACT

The research presented herein is a step toward the realization of a prototype flexible fixtureless assembly workcell. One requirement of a fixtureless workcell is the ability to locate and grasp parts that have been presented in arbitrary poses. The Feature CMAC neural network has been shown to solve the 3-DOF pose-estimation problem based on visual information for simple target parts. Through the use of more advanced feature-extraction algorithms and improved methods of manipulation, the Feature CMAC's capability has been extended for use with an unmodified industrial target part. The RMS positional error was found to be below 1.0 mm, and below 1.2° in orientation. In addition, the necessary tasks for application of the Feature CMAC pose-estimation system to a fixtureless assembly workcell have been identified.
ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Gabriele D’Eleuterio, for his insight and support in the completion of this thesis, and also the members of my research assessment committee, Dr. Chris Damaren and Dr. Robert Zee. Alanis’ modified gripper fingers were generously supplied by Dr. Gary Bone. This research was made possible by grants from the Institute for Robotics and Intelligent Systems (IRIS), and the Natural Sciences and Engineering Research Council of Canada (NSERC).

Thanks are also due to Joe Carusone for introducing me to the FCMAC and to Thierry Cherpillod for technical support. My labmates (in particular, Tim, Ernie, and Cathy) are to be commended for tolerating my “vocal programming” style.

Finally, my undying gratitude is extended to my parents, my girlfriend, and my God (not necessarily in that order) for their boundless love and support.
# TABLE OF CONTENTS

Abstract ......................................................................................................................... ii
Acknowledgements ........................................................................................................ iii
Table of Contents ........................................................................................................... iv
List of Tables .................................................................................................................. vi
List of Figures ................................................................................................................ vii
List of Abbreviations ..................................................................................................... ix
List of Symbols .............................................................................................................. x

1 Introduction ................................................................................................................. 1
  1.1 Motivation ................................................................................................................ 1
  1.2 The Flexible Fixtureless Assembly Workcell Project .............................................. 1
  1.3 Thesis Objective .................................................................................................... 3

2 Background .................................................................................................................. 5
  2.1 Feature Detection ................................................................................................... 5
      2.1.1 Boundary-based Methods .......................................................................... 7
      2.1.2 Local Search Methods .............................................................................. 7
      2.1.3 Other Feature Techniques ........................................................................ 9
  2.2 Pose-Estimation ....................................................................................................... 9
      2.2.1 Correspondence methods ........................................................................... 9
      2.2.2 Learning methods ...................................................................................... 12
  2.3 The Feature CMAC ............................................................................................... 13
      2.3.1 CMAC Properties ..................................................................................... 13
      2.3.2 The Feature CMAC .................................................................................. 17

3 Previous Research ...................................................................................................... 22
  3.1 Experimental Setup ............................................................................................... 22
  3.2 Previous Task ......................................................................................................... 23
      3.2.1 The Lego Target Part .............................................................................. 24
      3.2.2 Simple Feature Detection Algorithm ....................................................... 24
      3.2.3 Experimental Results ............................................................................... 25

4 Current Research ....................................................................................................... 28
4.1 Feature Detection ............................................................................................................. 28
  4.1.1 SUSAN Algorithm ........................................................................................................ 29
    4.1.1.1 Theory of the SUSAN Algorithm ........................................................................ 29
    4.1.1.2 Implementation of the SUSAN algorithm ....................................................... 31
  4.1.2 Parametric Feature Manifolds ..................................................................................... 34
    4.1.2.1 Theory of the PFM algorithm ............................................................................ 34
    4.1.2.2 Implementation of the PFM Algorithm ............................................................ 37
  4.1.3 Parameter Selection for Image Processing ................................................................. 41

4.2 Manipulation Issues ......................................................................................................... 44
  4.2.1 Kinematic Control ....................................................................................................... 44
  4.2.2 Grasping the Part ......................................................................................................... 49

5  Experiments and Results ................................................................................................... 52
  5.1 Pitched Lego Part .......................................................................................................... 52
  5.2 The Industrial Part ........................................................................................................ 53
    5.2.1 Feature Parameter Selection ................................................................................. 53
    5.2.2 FCMAC Experiments ............................................................................................. 55

6  Discussion ........................................................................................................................ 59
  6.1 The Pitched Lego Part .................................................................................................. 59
  6.2 The Industrial Part ........................................................................................................ 60
    6.2.1 Feature Parameter Selection ................................................................................. 60
    6.2.2 Discussion of Simple Results ................................................................................. 60
    6.2.3 Discussion of SUSAN Results ............................................................................... 61
    6.2.4 PFM Results ........................................................................................................... 63
    6.2.5 General Discussion ................................................................................................. 68

7  Conclusions and Suggestions ........................................................................................... 69
  7.1 Conclusions ................................................................................................................. 69
  7.2 Suggestions for Further Research ................................................................................ 71

References ........................................................................................................................... 73

Appendix A – Simple Algorithm Results ............................................................................ 78

Appendix B – SUSAN Algorithm Results ........................................................................... 84

Appendix C – PFM Algorithm Results ................................................................................ 93

Appendix D – PFM Algorithm Speed Improvements ....................................................... 103
  D.1 Rejection ...................................................................................................................... 103
  D.2 Coarse-to-Fine Algorithm ........................................................................................... 103
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 3-1</td>
<td>Results of experiments using the flat Lego target part</td>
<td>25</td>
</tr>
<tr>
<td>Table 4-1</td>
<td>Common parameter ranges for manifold sampling</td>
<td>40</td>
</tr>
<tr>
<td>Table 5-1</td>
<td>Results of experiments using the pitched Lego target part</td>
<td>52</td>
</tr>
<tr>
<td>Table 5-2</td>
<td>Feature cost analysis parameters and their best values</td>
<td>54</td>
</tr>
<tr>
<td>Table 5-3</td>
<td>Workspace for the industrial target part experiments</td>
<td>56</td>
</tr>
<tr>
<td>Table 5-4</td>
<td>The best experimental results for the industrial target part</td>
<td>58</td>
</tr>
<tr>
<td>Table 7-1</td>
<td>Numerical conclusions</td>
<td>69</td>
</tr>
<tr>
<td>Table A-1</td>
<td>List of the Simple parameter sets</td>
<td>78</td>
</tr>
<tr>
<td>Table B-1</td>
<td>List of the SUSAN parameter sets</td>
<td>84</td>
</tr>
<tr>
<td>Table C-1</td>
<td>List of the PFM parameter sets</td>
<td>93</td>
</tr>
<tr>
<td>Table C-2</td>
<td>PFM CMACs adjusted for orientation loss</td>
<td>93</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>Figure 1-1</td>
<td>An image of a fixtured workcell</td>
<td>1</td>
</tr>
<tr>
<td>Figure 1-2</td>
<td>Conceptual drawing of a fixtureless robotic workcell</td>
<td>1</td>
</tr>
<tr>
<td>Figure 1-3</td>
<td>The component parts of the IRIS &quot;strawman&quot; assembly task</td>
<td>2</td>
</tr>
<tr>
<td>Figure 1-4</td>
<td>Functional block diagram of the IRIS FFW</td>
<td>2</td>
</tr>
<tr>
<td>Figure 1-5</td>
<td>The 3-DOF pose-estimation problem</td>
<td>3</td>
</tr>
<tr>
<td>Figure 2-1</td>
<td>A typical visual pose-estimation system</td>
<td>5</td>
</tr>
<tr>
<td>Figure 2-2</td>
<td>The Trajkovic and Hedley detector</td>
<td>8</td>
</tr>
<tr>
<td>Figure 2-3</td>
<td>Feature quadrangles and tetrahedra</td>
<td>10</td>
</tr>
<tr>
<td>Figure 2-4</td>
<td>Foveal-peripheral segmentation in the PNN</td>
<td>12</td>
</tr>
<tr>
<td>Figure 2-5</td>
<td>CMAC quantization</td>
<td>14</td>
</tr>
<tr>
<td>Figure 2-6</td>
<td>Tiling shift in the CMAC</td>
<td>14</td>
</tr>
<tr>
<td>Figure 2-7</td>
<td>The CMAC neural network architecture</td>
<td>15</td>
</tr>
<tr>
<td>Figure 2-8</td>
<td>The Feature CMAC neural network architecture</td>
<td>18</td>
</tr>
<tr>
<td>Figure 3-1</td>
<td>Image of the experimental setup</td>
<td>22</td>
</tr>
<tr>
<td>Figure 3-2</td>
<td>Functional block diagram of the experimental setup</td>
<td>22</td>
</tr>
<tr>
<td>Figure 3-3</td>
<td>The CRS A465 6-DOF manipulator</td>
<td>23</td>
</tr>
<tr>
<td>Figure 3-4</td>
<td>The world coordinate frame</td>
<td>23</td>
</tr>
<tr>
<td>Figure 3-5</td>
<td>The original Lego target part</td>
<td>24</td>
</tr>
<tr>
<td>Figure 3-6</td>
<td>The Simple detector</td>
<td>24</td>
</tr>
<tr>
<td>Figure 3-7</td>
<td>Results of Simple feature-extraction</td>
<td>25</td>
</tr>
<tr>
<td>Figure 3-8</td>
<td>Error histograms for the original Lego target part</td>
<td>26</td>
</tr>
<tr>
<td>Figure 3-9</td>
<td>Pictorial explanation of bias errors</td>
<td>26</td>
</tr>
<tr>
<td>Figure 3-10</td>
<td>Toothpaste cartons used for recognition trials</td>
<td>27</td>
</tr>
<tr>
<td>Figure 4-1</td>
<td>The industrial target part</td>
<td>28</td>
</tr>
<tr>
<td>Figure 4-2</td>
<td>Too few/too many features</td>
<td>28</td>
</tr>
<tr>
<td>Figure 4-3</td>
<td>Results of Simple detection on the metal target part</td>
<td>29</td>
</tr>
<tr>
<td>Figure 4-4</td>
<td>The SUSAN detector</td>
<td>30</td>
</tr>
<tr>
<td>Figure 4-5</td>
<td>SUSAN comparison functions</td>
<td>30</td>
</tr>
<tr>
<td>Figure 4-6</td>
<td>Discrete circular windows</td>
<td>32</td>
</tr>
<tr>
<td>Figure 4-7</td>
<td>The SUSAN contiguity checking scheme</td>
<td>33</td>
</tr>
<tr>
<td>Figure 4-8</td>
<td>Results of SUSAN detection on the metal target part</td>
<td>33</td>
</tr>
<tr>
<td>Figure 4-9</td>
<td>The features used in [Baker et al., 1998]</td>
<td>34</td>
</tr>
<tr>
<td>Figure 4-10</td>
<td>Window-to-vector mapping scheme</td>
<td>35</td>
</tr>
<tr>
<td>Figure 4-11</td>
<td>The corner feature</td>
<td>37</td>
</tr>
<tr>
<td>Figure 4-12</td>
<td>The circular disc feature</td>
<td>39</td>
</tr>
<tr>
<td>Figure 4-13</td>
<td>Results of PFM detection on the metal target part</td>
<td>41</td>
</tr>
<tr>
<td>Figure 4-14</td>
<td>Pictorial example of feature cost</td>
<td>42</td>
</tr>
<tr>
<td>Figure 4-15</td>
<td>Training the TRN to find cluster centres</td>
<td>42</td>
</tr>
<tr>
<td>Figure 4-16</td>
<td>Typical outcome of feature cost training</td>
<td>44</td>
</tr>
<tr>
<td>Figure 4-17</td>
<td>The importance of rotation about the correct axis</td>
<td>45</td>
</tr>
<tr>
<td>Figure 4-18</td>
<td>The tool flange frame of reference</td>
<td>45</td>
</tr>
<tr>
<td>Figure 4-19</td>
<td>Trivial manipulation of the flat Lego target part</td>
<td>46</td>
</tr>
<tr>
<td>Figure 4-20</td>
<td>The pitched Lego target part</td>
<td>46</td>
</tr>
<tr>
<td>Figure 4-21</td>
<td>Nontrivial manipulation of the pitched Lego target part</td>
<td>47</td>
</tr>
<tr>
<td>Figure 4-22</td>
<td>Calibration of the tool transform</td>
<td>47</td>
</tr>
<tr>
<td>Figure 4-23</td>
<td>Rotation about the world x-axis</td>
<td>48</td>
</tr>
<tr>
<td>Figure 4-24</td>
<td>The standard parallel-jaw servogripper</td>
<td>49</td>
</tr>
<tr>
<td>Figure 4-25</td>
<td>The modified gripper</td>
<td>50</td>
</tr>
<tr>
<td>Figure 4-26</td>
<td>Limited mobility grasping</td>
<td>50</td>
</tr>
<tr>
<td>Figure 4-27</td>
<td>The worn edges at the grasp points</td>
<td>51</td>
</tr>
<tr>
<td>Figure 5-1</td>
<td>Error histograms for the pitched Lego target part</td>
<td>53</td>
</tr>
<tr>
<td>Figure 5-2</td>
<td>Images for the feature cost analysis</td>
<td>53</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5-3</td>
<td>Feature cost results for the Simple algorithm</td>
<td>54</td>
</tr>
<tr>
<td>5-4</td>
<td>Feature cost results for the SUSAN algorithm</td>
<td>55</td>
</tr>
<tr>
<td>5-5</td>
<td>Feature cost results for the PFM algorithm</td>
<td>55</td>
</tr>
<tr>
<td>6-1</td>
<td>Orientation loss</td>
<td>59</td>
</tr>
<tr>
<td>6-2</td>
<td>Preventing orientation loss</td>
<td>59</td>
</tr>
<tr>
<td>6-3</td>
<td>Grey-coded distance to manifold images</td>
<td>63</td>
</tr>
<tr>
<td>6-4</td>
<td>Decay of the K-L residue for varying window radius</td>
<td>63</td>
</tr>
<tr>
<td>6-5</td>
<td>Coarse pixel activation for various quantization intervals</td>
<td>64</td>
</tr>
<tr>
<td>6-6</td>
<td>Scatter plot showing regions where orientation was lost</td>
<td>65</td>
</tr>
<tr>
<td>6-7</td>
<td>Orientation loss of the metal target part</td>
<td>66</td>
</tr>
<tr>
<td>7-1</td>
<td>Necessary tasks for the prototype fixtureless workcell</td>
<td>71</td>
</tr>
<tr>
<td>A-1</td>
<td>Simple error by parameter set</td>
<td>78</td>
</tr>
<tr>
<td>A-2</td>
<td>Simple E1 vs. number of layers</td>
<td>79</td>
</tr>
<tr>
<td>A-3</td>
<td>Simple E2 vs. number of layers</td>
<td>80</td>
</tr>
<tr>
<td>A-4</td>
<td>Simple processing times by parameter set</td>
<td>80</td>
</tr>
<tr>
<td>A-5</td>
<td>Simple training time vs. number of layers</td>
<td>81</td>
</tr>
<tr>
<td>A-6</td>
<td>Simple testing time vs. number of layers</td>
<td>81</td>
</tr>
<tr>
<td>A-7</td>
<td>Simple FCMAC file size vs. number of layers</td>
<td>82</td>
</tr>
<tr>
<td>A-8</td>
<td>Error histograms for Simple FCMAC M4-63</td>
<td>83</td>
</tr>
<tr>
<td>B-1</td>
<td>SUSAN error by parameter set</td>
<td>84</td>
</tr>
<tr>
<td>B-2</td>
<td>SUSAN E1 vs. number of layers</td>
<td>85</td>
</tr>
<tr>
<td>B-3</td>
<td>SUSAN E2 vs. number of layers</td>
<td>85</td>
</tr>
<tr>
<td>B-4</td>
<td>SUSAN processing times by parameter set</td>
<td>86</td>
</tr>
<tr>
<td>B-5</td>
<td>SUSAN training time vs. number of layers</td>
<td>86</td>
</tr>
<tr>
<td>B-6</td>
<td>SUSAN testing time vs. number of layers</td>
<td>87</td>
</tr>
<tr>
<td>B-7</td>
<td>Error histograms for SUSAN FCMAC S2-63</td>
<td>88</td>
</tr>
<tr>
<td>B-8</td>
<td>SUSAN error vs. effective radius</td>
<td>89</td>
</tr>
<tr>
<td>B-9</td>
<td>SUSAN processing time vs. radius</td>
<td>89</td>
</tr>
<tr>
<td>B-10</td>
<td>SUSAN error vs. comparison parameter</td>
<td>90</td>
</tr>
<tr>
<td>B-11</td>
<td>SUSAN error vs. geometric threshold</td>
<td>90</td>
</tr>
<tr>
<td>B-12</td>
<td>SUSAN error vs. centre-of-mass threshold</td>
<td>91</td>
</tr>
<tr>
<td>B-13</td>
<td>SUSAN FCMAC file size vs. number of layers</td>
<td>92</td>
</tr>
<tr>
<td>C-1</td>
<td>PFM error vs. number of layers</td>
<td>94</td>
</tr>
<tr>
<td>C-2</td>
<td>PFM error by parameter set</td>
<td>94</td>
</tr>
<tr>
<td>C-3</td>
<td>PFM E1 vs. number of layers</td>
<td>95</td>
</tr>
<tr>
<td>C-4</td>
<td>PFM E2 vs. number of layers</td>
<td>95</td>
</tr>
<tr>
<td>C-5</td>
<td>PFM processing times by parameter set</td>
<td>96</td>
</tr>
<tr>
<td>C-6</td>
<td>PFM training time vs. number of layers</td>
<td>96</td>
</tr>
<tr>
<td>C-7</td>
<td>PFM testing time vs. number of layers</td>
<td>97</td>
</tr>
<tr>
<td>C-8</td>
<td>Error histograms for PFM FCMAC P0-63</td>
<td>98</td>
</tr>
<tr>
<td>C-9</td>
<td>PFM error vs. effective radius by binning</td>
<td>99</td>
</tr>
<tr>
<td>C-10</td>
<td>PFM error vs. effective radius by manifold</td>
<td>99</td>
</tr>
<tr>
<td>C-11</td>
<td>PFM error vs. magnitude threshold</td>
<td>100</td>
</tr>
<tr>
<td>C-12</td>
<td>PFM error vs. distance threshold</td>
<td>100</td>
</tr>
<tr>
<td>C-13</td>
<td>PFM error vs. check range</td>
<td>101</td>
</tr>
<tr>
<td>C-14</td>
<td>PFM error vs. sampling density</td>
<td>101</td>
</tr>
<tr>
<td>C-15</td>
<td>PFM FCMAC file size vs. number of layers</td>
<td>102</td>
</tr>
<tr>
<td>D-1</td>
<td>Structure of a manifold file</td>
<td>104</td>
</tr>
<tr>
<td>D-2</td>
<td>Trinary search tree formation</td>
<td>105</td>
</tr>
<tr>
<td>D-3</td>
<td>Search example</td>
<td>105</td>
</tr>
<tr>
<td>D-4</td>
<td>Alternate search schemes</td>
<td>106</td>
</tr>
</tbody>
</table>
# LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-D</td>
<td>Two-dimensional</td>
</tr>
<tr>
<td>3-D</td>
<td>Three-dimensional</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>ARO</td>
<td>Adaptive Response Organizer</td>
</tr>
<tr>
<td>CAD</td>
<td>Computer Aided Design</td>
</tr>
<tr>
<td>CCD</td>
<td>Charge-Coupled Device</td>
</tr>
<tr>
<td>CEGI</td>
<td>Complex Extended Gaussian Image</td>
</tr>
<tr>
<td>CMAC</td>
<td>Cerebellar Model Articulation Controller, or Cerebellar Model Arithmetic Computer</td>
</tr>
<tr>
<td>CSS</td>
<td>Curvature Scale Space</td>
</tr>
<tr>
<td>DDE</td>
<td>Dynamic Data Exchange</td>
</tr>
<tr>
<td>FCMAC</td>
<td>Feature CMAC</td>
</tr>
<tr>
<td>DOF</td>
<td>Degree of Freedom</td>
</tr>
<tr>
<td>FFWA</td>
<td>Flexible Fixtureless Assembly Workcell</td>
</tr>
<tr>
<td>IRIS</td>
<td>Institute for Robotics and Intelligent Systems</td>
</tr>
<tr>
<td>GMNN</td>
<td>General Memory Neural Network</td>
</tr>
<tr>
<td>K-L</td>
<td>Karhunen-Loéve (transform)</td>
</tr>
<tr>
<td>LMS</td>
<td>Least Mean Square</td>
</tr>
<tr>
<td>NT</td>
<td>New Technology</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PFM</td>
<td>Parametric Feature Manifold</td>
</tr>
<tr>
<td>PNN</td>
<td>Position Neural Network</td>
</tr>
<tr>
<td>RAM</td>
<td>Random-Access Memory</td>
</tr>
<tr>
<td>RMS</td>
<td>Root Mean Squared</td>
</tr>
<tr>
<td>SCAP</td>
<td>Synegetic Computer using Adjoint Prototypes</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
<tr>
<td>SUSAN</td>
<td>Single Unvalue Segment Assimilating Nucleus</td>
</tr>
<tr>
<td>TRN</td>
<td>Topology Representing Network</td>
</tr>
<tr>
<td>USAN</td>
<td>Unvalue Segment Assimilating Nucleus</td>
</tr>
<tr>
<td>UTIAS</td>
<td>University of Toronto Institute for Aerospace Studies</td>
</tr>
</tbody>
</table>
# LIST OF SYMBOLS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>Image intensity constant</td>
</tr>
<tr>
<td>$b$</td>
<td>Image sampling binning factor</td>
</tr>
<tr>
<td>$B$</td>
<td>Image intensity constant</td>
</tr>
<tr>
<td>$c$</td>
<td>Number of layer pattern neuron connections</td>
</tr>
<tr>
<td></td>
<td>SUSAN comparison function</td>
</tr>
<tr>
<td>$C$</td>
<td>CMAC quantization interval</td>
</tr>
<tr>
<td></td>
<td>Feature cost</td>
</tr>
<tr>
<td>$C$</td>
<td>Rotation matrix</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>TRN learning rate</td>
</tr>
<tr>
<td>$D$</td>
<td>Distance contribution to feature cost</td>
</tr>
<tr>
<td>$e$</td>
<td>Single composite error</td>
</tr>
<tr>
<td>$e$</td>
<td>Eigenvector</td>
</tr>
<tr>
<td>$E_1$</td>
<td>Raw average composite error</td>
</tr>
<tr>
<td>$E_2$</td>
<td>Adjusted average composite error</td>
</tr>
<tr>
<td>$\varepsilon$</td>
<td>TRN learning multiplier</td>
</tr>
<tr>
<td>$g$</td>
<td>SUSAN geometric threshold</td>
</tr>
<tr>
<td>$i$</td>
<td>Image coordinate</td>
</tr>
<tr>
<td>$j$</td>
<td>Image coordinate</td>
</tr>
<tr>
<td>$l$</td>
<td>FCMAC image pattern neuron activation</td>
</tr>
<tr>
<td></td>
<td>Image intensity</td>
</tr>
<tr>
<td>$l$</td>
<td>Characteristic length of target part</td>
</tr>
<tr>
<td>$L$</td>
<td>FCMAC layer pattern neuron activation</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>TRN ranking factor</td>
</tr>
<tr>
<td></td>
<td>Eigenvector</td>
</tr>
<tr>
<td>$m$</td>
<td>SUSAN centre-of-mass threshold</td>
</tr>
<tr>
<td>$M$</td>
<td>Number of TRN neurons</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Image patch mean value</td>
</tr>
<tr>
<td>$n$</td>
<td>USAN area</td>
</tr>
<tr>
<td>$\nu$</td>
<td>Image patch magnitude</td>
</tr>
<tr>
<td>$p$</td>
<td>Intermediate distance parameter</td>
</tr>
<tr>
<td>$P$</td>
<td>Pitch angle</td>
</tr>
<tr>
<td>$P$</td>
<td>FCMAC matrix of pose coordinates</td>
</tr>
<tr>
<td>$q$</td>
<td>Matrix of generalized feature parameters</td>
</tr>
<tr>
<td>$r$</td>
<td>Radius parameter</td>
</tr>
<tr>
<td>$r$</td>
<td>Matrix of image coordinates</td>
</tr>
<tr>
<td>Symbol</td>
<td>Definition</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
</tr>
<tr>
<td>$R$</td>
<td>SUSAN edge response</td>
</tr>
<tr>
<td></td>
<td>Karhunen-Loève residue</td>
</tr>
<tr>
<td>$R_s$</td>
<td>Image patch autocorrelation matrix</td>
</tr>
<tr>
<td>$s$</td>
<td>CMAC tiling shift</td>
</tr>
<tr>
<td>$t$</td>
<td>SUSAN comparison threshold</td>
</tr>
<tr>
<td>$T$</td>
<td>Number of TRN training steps</td>
</tr>
<tr>
<td></td>
<td>Twist angle</td>
</tr>
<tr>
<td>$\theta$</td>
<td>Orientation parameter</td>
</tr>
<tr>
<td>$\theta_2$</td>
<td>Subtended angle parameter</td>
</tr>
<tr>
<td>$u$</td>
<td>Heavyside step function</td>
</tr>
<tr>
<td>$w$</td>
<td>CMAC weight matrix</td>
</tr>
<tr>
<td></td>
<td>TRN neuron locations</td>
</tr>
<tr>
<td>$W$</td>
<td>Set of pixels in a discrete window</td>
</tr>
<tr>
<td>$x$</td>
<td>World distance coordinate</td>
</tr>
<tr>
<td>$x'$</td>
<td>CMAC neuron activation matrix</td>
</tr>
<tr>
<td>$y$</td>
<td>World distance coordinate</td>
</tr>
<tr>
<td>$z$</td>
<td>World distance coordinate</td>
</tr>
<tr>
<td></td>
<td>Intermediate distance parameter</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

1.1 Motivation

The archetype of fixtureless assembly is currently receiving attention in many areas of industrial manufacturing. In a traditional robotic assembly workcell, one or more fixtures are required to position, localize and mate parts before a robot can join them. Figure 1-1 demonstrates the sheer bulk of a fixture compared to the size of the part and the robot. Dependence on fixtures is a serious detriment to flexible manufacturing since these expensive, custom-made jigs must be scrapped and replaced whenever one wishes to change parts or processes. This turnover necessitates a long lead time for the design of new fixtures, recurring costs in their manufacture, and down-time on the factory floor as they are swapped into the assembly line. It is estimated that the total recurring cost of assembly fixtures in the automotive industry is on the order of US$ 10M per plant per year [Mills, 2000]. In an effort to reduce recurring costs and increase flexibility in the industrial sector, research is being done to allow removal of fixtures completely from the factory floor. The concept of a fixtureless workcell replaces fixtures with reconfigurable robots that are trained to localize and mate parts for assembly in conjunction with the traditional mating robot. A one-time cost is incurred to set up the workcell; thereafter, the robots are simply reprogrammed for changes in parts and processes. Figure 1-2 shows a concept diagram of the fixtureless robotic workcell.

1.2 The Flexible Fixtureless Assembly Workcell Project

In 1997, researchers from six research groups in four universities, under the auspices of the Institute for Robotics and Intelligent Systems (IRIS), began working toward a Robust, Autonomous, Intelligent, Flexible, Fixtureless Assembly Workcell [D'Eleuterio et al., 1997]. The goals of the team were, and still are, to develop the technology necessary to make fixtureless assembly viable, and to integrate
these technologies into a working prototype Flexible Fixtureless Assembly Workcell (FFAW). The workcell will demonstrate the following key performance features:

- **Robust**: The system will be robust to uncertainties such as part misalignment and deformation.
- **Autonomous**: The system will require very little human interaction and supervision.
- **Intelligent**: The system will make decisions based on information it senses from the workspace.
- **Flexible**: The systems will be easily reconfigurable for several assembly operations and collections of parts.
- **Fixtureless**: The system will contain no specialized devices to hold or localize the parts.

Fixtureless assembly is an inherently difficult task encompassing many problem areas in robotics such as precision control, part mating, intelligent grasping, and sensor integration. Key technologies will have to be invented and developed in order to assemble components with sufficient accuracy. As an additional consideration, the system must prove cheaper over its entire life cycle than the traditional (and occasionally brute-force) methods currently being used in order to be economically viable.

The end goal of the IRIS team is to construct a fully functional proof-of-concept prototype fixtureless assembly workcell. To that end, a representative task (colloquially known as a "strawman" task) has been chosen, which is to weld together the five steel pieces that comprise the front wing panel of a Chevrolet Lumina (shown in Figure 1-3).

A functional block diagram of the workcell is shown in Figure 1-4. Each block corresponds to the work of one of the six university groups.

![Figure 1-3: The component parts of the IRIS "strawman" assembly task.](image)

![Figure 1-4: A functional block diagram of the IRIS prototype fixtureless assembly workcell.](image)
• **Find the Parts:** Since there will be no fixtures, parts will not be presented in prelocalized poses. As such, a vision system will be needed to determine the initial positions and orientations of the parts so that the robots may acquire an initial grasp.

• **Grip the Parts:** A single reconfigurable gripper will be needed in order to successfully manipulate a wide class of objects. No alterations will be made to the part and gripping will not damage the part.

• **Localize the Parts:** In order to obtain precise positioning and mating of the parts, a sensor suite will be needed to extract the correct six degree-of-freedom pose of the gripped part.

• **Align the Parts:** A visual servoing system will be required in order to mate two parts or subassemblies that must be joined.

• **Join the Parts:** A laser welder with a precise seam tracking system will be used to join any two parts or subassemblies.

• **Control the System:** A virtual environment for task planning, refinement and scheduling will be needed in order to visualize and troubleshoot the assembly process before it is implemented with the actual robotic hardware.

### 1.3 Thesis Objective

The “Find the Parts” functional block of the FFAW is the responsibility of the Space Robotics Group at the University of Toronto Institute for Aerospace Studies (UTIAS). The problem under discussion in this thesis is that of machine vision for automatic target pose-estimation. When implemented, this vision system will empower a robot to recognize, to servo to, and to grasp a part that has been placed in an unstructured manner within its workspace.

True pose-estimation is a six degree-of-freedom (DOF) problem in which the full position and orientation of a target object are determined. However, fixtureless assembly can still be performed if the problem is reasonably restricted to rigid parts constrained to slide in a plane. For example, imagine a human operator casually placing parts face-up on a table.

![Figure 1-5: The three degree-of-freedom pose-estimation problem.](image-url)
The problem now becomes one of finding the \((x,y)\) position of the object and its rotational orientation about the \(z\)-axis (see Figure 1-5).

The pose-estimation method described in this paper uses an Artificial Neural Network (ANN) called the Feature CMAC (FCMAC). CMAC is, itself, the acronym for another ANN, explained in detail in Section 2.3. Features detected in the image are transformed into pose data to be used as set points for a conventional robotic controller. This allows the robot to find and grasp any part that the FCMAC has been trained to locate.

Section 2 provides background technical information in the form of literature reviews of image processing and machine vision-based pose-estimation schemes. Additionally, the theoretical principles behind the ANN vision system for the workcell are presented. The experimental setup and previous work on pose-estimation are presented in Section 3. Section 4 details the new developments presented in this thesis, both in the areas of feature detection and grasp procedures for an industrial part. Experimental details and their results are presented in Section 5. Section 6 includes a discussion of the results and interesting observations arising out of the experiments. Finally, Section 7 states the conclusions of this thesis and makes suggestions for future research.
2 BACKGROUND

The functional structure of a typical visual pose-estimation solution is shown in Figure 2-1. A camera (or set of cameras) observe the workspace in which the target object will be found. Some form of image processing is then applied; typically a form of feature-extraction is used, but occasionally Fourier and optical transformations serve as alternatives. The results of the image processing step are used as the input to a pose-estimation algorithm, whose output is a set of coordinates that describe the position and orientation of the part in some relevant coordinate system.

![Figure 2-1: Functional block diagram of a typical visual pose-estimation system.](image)

Many processing algorithms can be used for the feature detection and pose-estimation functional blocks. A survey of the current literature in these two fields is included in the following two subsections.

2.1 Feature Detection

There is a vast ocean of literature on image processing, particularly in the field of feature detection. To provide a complete summary of all methods of feature-extraction would constitute a complete thesis in itself. In order to narrow the field of articles, the survey paid particular attention to the following criteria:

- **Fast:** It is desirable in manufacturing to complete operations as quickly as possible in order to maximize the output of marketable goods. By extension, the pose-estimation task should be as rapid as possible to keep the system from waiting. Since the FFAW will be implemented using commonly available computer hardware, the feature-extraction algorithm should be as computationally simple as possible.

- **Reliable:** The number and quality of features detected should have only the smallest dependence on the operating point of the system. Changes in lighting, temperature, and incremental changes in part positioning should result in feature patterns similar to those found during training.
• **Applicable:** The algorithm should look for features that are representative of an industrial part. For example, a hard corner detector is inappropriate for parts with large, sweeping curves.

• **Adaptable:** To keep the system flexible, the image processing should not be aimed solely at one class of parts. While some amount of tuning is desirable to maximize the accuracy of the pose-estimations, it should be simple enough that a minimum number of easily modifiable parameters are optimized.

As will be discussed in greater detail in Section 2.3, two-dimensional discontinuities in the image (also known as point features) are the best input for the FCMAC. As such, one-dimensional features such as edges and lines are not dealt with in this literature survey. Seminal articles in the field of edge detection include [Marr and Hildreth, 1980], and [Canny, 1986].

Corners are an excellent and well-researched point feature. However, a single, standard definition of what constitutes a “corner” does not exist. Some researchers define a corner as a sharp angle in the image; others as the point where a curve has a local maximum of curvature; still others even consider the somewhat existential definition of “something a human would call a corner”. A wide variety of detection methods have been postulated. For a very thorough review of corner detection literature, see [Mokhtarian and Suomela, 1998]. Some of the more interesting and representative articles on corner detection are described below.

The design and performance of feature detection algorithms relies greatly on the goals of the designer. Some look for fast, robust detectors; others want precision of localization; still others look for parameter recovery such as subtended angle, sharpness, etc. In the FCMAC application, precise, single-pixel corners do not cause the network to perform well. This ANN works best with a detector that finds locally clustered approximate corners (see Section 2.3).

Two themes continually recur in the field of corner detection. The first theme is a trade-off between boundary-based and local searching. Boundary-based techniques perform a segmentation of the object from its surroundings, then look for boundary locations where the curvature is a local minimum. Local searches apply a mathematical function (such as a filter) to the pixels found within a small window. The window is then passed over the entire space of the image.

The second theme contrasts gradient-based versus analytical-model-based processing. Both of these methods can be combined with boundary-based or local techniques. The gradients (i.e., directional derivatives) of an image are the key measure in the first method, since discontinuities in image intensity are a telltale sign of features. Alternatively, one can find the best fit mathematical model of a corner (such as a spline or parameterized curve) to a boundary or window of pixels. Best fit parameters yield the corner location for a boundary or the value of cornerness for a window.
2.1.1 Boundary-based Methods

Lee et al. [1993] use a boundary-based approach where corner points are found by applying a wavelet transform to the object boundary. Locations where the modulus of the wavelet-transformed image is a maximum correspond to sharp changes in edge orientation. These maximal points are classified as corners. The algorithm was tested successfully, but with synthetic images only.

Kang and Kweon [1998] examine the problem of extracting groups of lines that accurately describe an object. In so doing, they contribute to boundary-based corner detection by applying a heuristic algorithm that finds and completes missing corner junctions formed by two line segments that come close but do not intersect. A junction quality factor is derived to avoid detecting false corners. When applied to real images, imperfect corner junctions were successfully detected.

Other researchers in boundary-based feature detection have examined the importance of scale. Once a boundary has been found, examining it from a gross perspective can point out clear corners which can then be examined at successively finer scales to localize the precise positions of features. Mokhtarian and Suomela [1998] and Fidrich and Thirion [1998] both perform this type of analysis. In the former a curvature scale space (CSS) detector calculates the curvature from the scale-altered image intensities to find local maxima. Their technique was applied to real and synthetic images, and was shown to be more robust to noise than the Single Univalue Segment Assimilating Nucleus (SUSAN) technique [Smith and Brady, 1997] and the Plessey detector [Harris and Stephens, 1988]. Fidrich and Thirion examine an image's extreme values of curvature. Corners are marked at points where the extrema change sign as scale is varied. Essentially, this identifies corners as points of inflection in the boundary extrema when expressed in scale space. Their work is applied to the registration of medical imagery. Lindeberg [1998] proposes a method for finding appropriate scales for image processing; scales that still contain useful information, but abstract away smaller variations that may cause feature detectors to fail. He forms a trial function, out of normalized image derivatives, that peaks at certain scales. These peaks, he reasons, are due to the scale value being a characteristic length of the image data structure. The technique is applied to real and synthetic images.

2.1.2 Local Search Methods

Harris and Stephens [1988] extend the Moravec interest operator to create a combined edge and corner finder commonly referred to as the Plessey detector. A second-order Taylor expansion of the local autocorrelation matrix leads to an elegant formula for corner response; the response is positive for corners, negative for edges, and small for nonFeatured regions. The combined detector is applied successfully to real images for frame-to-frame correspondence.

Wang and Brady [1995] use a classic gradient-based approach by noting two things: first, that the total curvature of the image intensity is proportional to the second-order derivative taken in the direction of the edge normal, and second, that the total curvature is inversely proportional to the magnitude of the edge
normal. Since the former can be thought of as the cornerness measure, and the latter as an indication of false positivity, the overall corner measure is defined as the difference between the gradient value and the magnitude value. Their numerical method uses linear interpolation to compute the image derivatives, resulting in a subpixel estimate of corner location. The method was applied favourably to real images.

![Figure 2-2: The Trajkovic and Hedley detector. A) The circular detector window with rotating line PP'. B) Within an object, both P and P' fall within the USAN (shown in black) for several orientations. C) At an edge, P and P' fall within the USAN for only one orientation. D) At a corner, at least one of P and P' must fall outside of the USAN.]

Trajkovic and Hedley [1998] begin with the axiom that the intensity should change rapidly in all directions from a corner since it is a 2-D discontinuity. They pass a circular window over the image and find the region within that has a similar intensity to that of the centre pixel, or “nucleus”. This region is referred to as the Univalence Segment Assimilating Nucleus (USAN) (see Section 4.1.1, for greater detail). One draws a line through the nucleus, resulting in two points P and P' on the boundary of the window (see Figure 2-2). If, for any rotation of the line PP', at most one of P and P' falls within the USAN, then a corner has been detected. This method performed well overall when compared to the SUSAN [Smith and Brady, 1997], Plessey [Harris and Stephens, 1988], and Wang [1995] algorithms.

Ando [2000] also searches for 2-D discontinuities in the image intensity. He does so by applying a statistical analysis to a small window of the image, in order to find its gradient distribution. The homogeneity operator and its complement are derived and shown to indicate regions of the image where the gradient varies in all directions, and unidirectionally, respectively. These operators are tested successfully with real images.

Departing from the realm of mathematical models, Dias et al. [1995] use a three layer connectionist ANN to detect corners within a small image window. The network is trained with supervision on binary synthetic corner images with subtended angles in 45° increments. Image windows are classified in the final layer as either “corner” or “not corner”. Their experimental data do not inspire confidence, since the network found only 71% of imperfect corners in a set of very simple binary synthetic images; however, the results would clearly improve with a larger network and more thorough training.
2.1.3 Other Feature Techniques

Other interesting papers in feature detection include a method of finding ellipses using a Hough transformation [Bennett and Burridge, 1999]. The method returns the area and centre position of an ellipse, which could in turn be used as a point feature for the FCMAC ANN. Rosin [1996] proposes augmented descriptions for corners, based on several of their parameters (subtended angle, sharpness, T-junction, etc.). If successful, this technique could divide corners into different feature classes that could be discriminated by an FCMAC trained using multiple feature layers (see Section 2.3.2).

Biologically inspired image processing is another fascinating topic. Navabi and Agarwal [1998] have developed the three layer Adaptive Response Organizer (ARO) ANN. Interlayer connections are used to create an antagonistic centre-surround detector. While their work constitutes an interesting model of retinal response in mammalian vision, they do not suggest how their network could be applied to digital image processing. Moving further down the vision processing pathway, Yadid-Pecht and Gur [1996] have developed a biologically inspired image classifier which has the ability to select which features from its base set are the best for classification. However, the authors use an arbitrary set of binary feature detectors, and as such little is contributed to the study of feature detection.

2.2 Pose-Estimation

There are two major approaches to solving the problem of pose-estimation from visual data. The first is to use an internal representation (for example, a CAD model) of the part and its visual features. A comparison can then be made between the observed features and the model to determine the most likely pose. These algorithms are collectively known as correspondence methods. The second class of methods use neural networks or other artificial learning algorithms. In these approaches, the system must go through a series of training cycles in order to capture a sufficient amount of information about the target object. An association or mapping is made between the detected features and the known poses. In both learning and correspondence methods the feature information is transformed into pose coordinates, so virtually any choice of image processing algorithm (as discussed in Section 2.1) can be used with any pose estimator.

2.2.1 Correspondence methods

Several model-matching algorithms operate on a local level by performing feature detection and then matching individual features between the model and the image. These techniques are generally more precise when given good input, but are more computationally intense and suffer greatly when features are mismatched. In contrast, an alternate set of algorithms use global properties such as centroids and moments of image intensities. While they are much more simple computationally, they rely heavily on segmentation algorithms and are non-robust to occlusion, particularly self-occlusion.
Lowe [1991] proposed a conceptually simple and generally applicable algorithm for determining both an object's pose relative to a camera, as well as the focal length of the camera lens. A projective transformation that maps model feature points into the camera coordinate system is assumed, and as such, all calculations can be performed in this frame of reference. Translation and rotation vectors are then defined based on the camera coordinates. Because of the assumed transformation, the image partial derivatives can be approximated, allowing the translation and rotation vectors to be solved via Newton's method. Three point correspondences each yield two linear equations (one in each of the image position directions), which can then be solved to find the corrections for the next iteration. The algorithm performed slowly but effectively with real images of a complex part. Araújo et al. [1998] use Lowe's algorithm with the assumption of a fully perspective transformation. Better approximations can be made for the partial derivatives, leading to an increase in accuracy. The algorithm performed well on a simple case using synthetic data. The difficulty with Lowe's formulation is the initial feature correspondence, and the choice of the best three features to generate the model fitting.

Another analytical method was developed by Abidi and Chandra [1995], which determines pose from a quadrangle of features. Using the focal point of the camera and three of the four feature points on the model, a tetrahedron can be defined. For each of the four possible tetrahedra (see Figure 2-3B), the volume is calculated. This is then mapped using a pinhole camera model into the image space, where it is matched with the four triangles formed by the feature quadrangle (see Figure 2-3A). This solution is closed form, unique, and very fast. However, it relies completely on finding the same four points in every image, and on the assumption of a pinhole model for the camera lens.

Beveridge and Riseman [1995] apply a local feature matching algorithm to the problem of robot navigation using real images. A full perspective algorithm is run to generate fine position estimates, with two weak perspective algorithms running in parallel for fast, incremental updates. A generate-and-test approach yields the best least-squares correspondence between the room model and the image.
Moving away from local point feature matching, Lee and Haralick [1996] use line correspondence to simultaneously estimate all six pose coordinates simultaneously. Their algorithm performs better than position/orientation decoupled methods, but relies on statistics and a noise model of the image. Chang and Tsai [1999] also use Lee and Haralick's method, but their major contribution is a statistical two-stage testing algorithm that evaluates the usefulness of the pose-estimation. In the first stage, the quality of the input lines themselves are tested against a set of thresholds. If there is insufficient data or if the lines are unclear, the input is scrapped. After passing the first stage, the pose is estimated. The second stage evaluates this output to see if it is sufficiently precise for a useful decision to be made. The full rejection algorithm was tested successfully with real and synthetic images.

Chen and Stockman [1996] predict a synthetic edge image for a curved model based on its pose parameters and a weak perspective transformation. This edge map is then iteratively matched to the image edge map until the closest estimate of the pose parameters are found. Occlusion is overcome by the use of heuristic matching rather than by examination of point features. The algorithm assumes image segmentation of the object from the background, and the majority of the computational complexity lies in generating the model's silhouette and the resulting edge image.

Basri and Jacobs [1997] attempt to compromise between global and local techniques by matching regions. Correspondences are found between the image and segmented, convex areas of the model. This decreases the complexity of matching that would be encountered using point features, but retains some of the insensitivity to occlusion. Their solution applies linear programming to the forward constraints of mapping the model to the image. The system works for affine transformations followed by either an orthographic or perspective projection. Jacobs and Basri [1999] look closer into the effects of occlusion, and tests the algorithm on real and synthetic images.

There are several interesting model-based pose-estimation techniques that do not involve classical correspondence, but should be briefly mentioned. Topological goniometry [Etquerra and Mullick, 1996] derives a full pose estimate from 3-D input data. The Hausdorff distance [Rucklidge, 1997] is a measure of the similarity between two flat shapes, and has been used for pose-estimation where the target object's depth is small compared to the distance from the camera. It is robust to excessive and erroneous features, multiple object instances, and partial occlusion. Surface normals of an object model can be described using the Complex Extended Gaussian Image (CEGI) [Kang and Ikeuchi, 1993]. In this complex formulation, the magnitude is proportional to the visible area of a surface, while the phase represents its distance from the image origin. This encoding allows the translation and orientation contributions to be decoupled into two, easier, least squares problems.
2.2.2 Learning methods

One of the seminal works on ANN-based vision is the neocognitron developed by Fukushima [1988]. The network is trained using unsupervised learning, and incorporates competition and backward connections in order to allow selective attention to parts of the input image. Each stage of the neocognitron is comprised of S-cells which learn characteristic features, and C-cells which provide shift-invariance. Fukushima applied the network to optical character recognition, but Hatakeyama and Kakazu [1995] expanded its usefulness to include translational position estimation. A tradeoff arises in the size of the C-cell receptive field; a larger field means greater shift intolerance (and hence a wider range of possible locations), but more stages will be required to get an accurate recognition of the target due to the increased blurring. The expanded neocognitron was applied to binary, synthetic images and with estimates accurate to single-pixel resolution.

Yagi et al. [1994] developed the three layer Position Neural Network (PNN). The input neurons perform a foveal-peripheral segmentation of the image (see Figure 2-4), which has the advantage of finding targets closest to the optical axis first. The inputs are fully connected to a 15-unit hidden layer, and thence to the output layer with neurons for the $x$ and $y$ location of the object. The authors use this network in a real-world active vision scenario where the camera's pan and tilt coordinates are successively refined until the object lies directly along the optical axis.

Hogg et al. [1995] use a two-layer network called the Synergetic Computer using Adjoint Prototypes (SCAP), whose goal is not high accuracy or minimal use of information, but rather minimal processing time based on the simplest information. The network is trained using mean-magnitude normalized images (see Section 4.1.2) of known poses of the target. From this information, a set of adjoint prototype images are formed that contain only the useful features of the part and reject background features found in every training image. Neurons in the internal layer perform a dot product between input images and these prototypes in order to arrive at an estimation of the target's orientation. The main drawback to the system is the difficulty of choosing appropriate nominal views. The system was tested on synthetic images with pure rotations about a single axis, with an overall accuracy of +/- 2.6 degrees.

Khotanzad and Liou [1996] use very simple banks of neural networks for target recognition and pose of military vehicles. The network operates on scale-normalized binary silhouettes to return the aspect and elevation of the viewing axis. Its primary drawback is its reliance on accurate image segmentation.
Other pose-estimation systems incorporate learning and model independence without the use of neural networks. Murase and Nayar [1995] use appearance based matching for recognition and pose-estimation, with a technique identical to their feature-extraction method described in Section 4.1.2. The Karhunen-Loéve (K-L) transform is applied to scale normalized images of the part to create a set of basis eigenvectors. The images are projected into this reduced-order eigenspace to form a manifold parameterized by the pose variables. Recall is performed by projecting a test image into the eigenspace and interpolating between the nearest training points. A universal eigenspace is used to perform the recognition task, while an object eigenspace provides accuracy for pose-estimation. The system was trained every four degrees about a single axis under several lighting conditions, and could recall with an accuracy of a single degree.

Ohba and Ikeuchi [1997] utilize the eigenwindow technique for finding the position of occluded objects. The system is trained by passing a small window over characteristic views of the target part and applying the K-L transform. The resulting windows act as basis functions to encode feature information about the target part. When applied to a test image, each eigenwindow adds a likely pose; by looking at the final cluster of suggested poses, a good 2-D location estimate can be calculated. Storing all of the windows is memory intensive, so an algorithm is suggested to store and use only those windows whose contributions are significant.

2.3 The Feature CMAC

The Feature CMAC (of FCMAC for short) will be discussed in greater detail since it is the only pose-estimation method used in this research. It is, in essence, an associative memory that maps patterns of features to a set of coordinates x, y, and θ.

2.3.1 CMAC Properties

Before examining the Feature CMAC, it is useful to first understand the principles which it borrows from the Albus perceptron (or, CMAC). The following theoretical explanation synthesizes the information found in [Albus, 1975] and [Miller et al., 1990]. The acronym CMAC is used to stand for either Cerebellar Model Articulation Controller, or Cerebellar Model Arithmetic Computer. Both descriptions are apt; “cerebellar model” refers to the network’s inspiration from the biological function of the human cerebellum. “Articulation controller” refers to the cerebellum’s control of human muscles, and hence is usually the term of choice in robotic control applications. The mathematical formulation of the network’s output is a simple arithmetic summation of memory units, which leads to the term “arithmetic computer”.
The CMAC is used to learn and generalize the output of a scalar multivariable nonlinear function. The function must be continuous and sufficiently smooth on closed intervals of the input variables. One can view the CMAC as an improved lookup table that simultaneously accesses several entries and calculates a final output based on their contributions. The key to the CMAC's ability to do so lies in the coarse quantization of its input variables. As an example, imagine a control function with two bounded, continuous inputs \( x \) and \( y \), and a scalar output \( z(x,y) \). The input domain would appear as in Figure 2-5A. It is axiomatic that for any form of digital computation to be done, each variable will have to be quantized, as shown in Figure 2-5B. The width of the quantization is known as the resolution of the system.\(^1\) If one wished, processing could be done at this fine resolution scale as is done in the case of a simple lookup table. However, the table would require an entry for every possible value of \((x,y)\).

A decrease in memory and a sharp decrease in training can be realized by sharing training information over its local region of the input space (the function must be sufficiently smooth and continuous for this to be viable). Information spreading is realized by coarsely quantizing the input variables. The number of resolvable values in a single coarse region is called the quantization interval, \( C \). The same coarse quantization is applied to all input variables, which has the effect of dividing the input domain into several hyperrectangles. In the 2-D example shown in Figure 2-5C, the result is a square. \( C \) of these coarsely quantized layers are generated and offset from each other. By having all these layers, the network still retains its original resolution. Figure 2-6A shows the standard diagonal offset method, which is commonly used due to its ease of calculation; however any offset pattern that retains the original

\(^1\) Note that the resolution does not have to be the same for each input variable; however for ease of visualization the figures in this thesis will be drawn with equal resolution.

---

**Figure 2-5**: CMAC quantization. A) A 2-D input domain. B) Fine quantization. C) Coarse quantization.

**Figure 2-6**: Tiling shift in the CMAC. A) The standard diagonal offset. B) A tiling shift of \([1,2]\). C) A tiling shift of \([1,3]\) (optimal for \( C = 8 \)). D) While it would never be used, a random tiling is still a valid option.
resolution is acceptable. For a more detailed analysis of offset alternatives and their effect on training, see [Parks and Militzer, 1989]. The tiling relationship optimal for information sharing in a 2-D input domain is [An et al., 1991]:

\[ C = s^2 - 1 \quad , \quad s \geq 2 \]  

(1)

where \( C \) is the number of layers and \( s \) is the tiling shift. The result of optimal tiling for \( C = 8 \) is shown in Figure 2-6C.

In future uses of the network, a given set of input coordinates will excite the \( C \) hyperrectangles that cover its area of the input domain. Higher \( C \) means better generalization since information is spread over a greater region of the input domain and requires less memory, but leads to less accuracy.

Once the coarsely quantized layers are established, training can commence. Each coarse hyperrectangle has a stored output value that it will contribute when activated. Learning usually takes the form of supervised modification of output values using the Least Mean Square (LMS) rule

\[ \Delta w = (\beta/C)(f_0 - w^T x_0)x_0 \]  

(2)

where \( \beta \) is the learning rate (or gain), \( f_0 \) is the desired output, \( x_0 \) is the vector of binary excitations for the coarse hyperrectangles, and \( w \) is the vector of weights for each hyperrectangle. The LMS rule can also be used with a gain of one in cases where training data is especially sparse. Normally this scheme is not used since a unity gain can lead to learning interference. Output value adjustment only takes place for those coarse hyperrectangles that cover the input values. Thompson and Kwon [1995] delve extensively into CMAC training techniques, and suggest the neighbourhood sequential and random training methods. In the former, one uses

---

2 Parks and Militzer [1989] used an exhaustive search to generate a table of optimal tiling offsets for domains up to 10-D, and quantization levels \( C \leq 50 \).
**a priori** knowledge of the coarse quantization scheme to pick a minimum number of points that will teach the CMAC over its entire input domain (and hence, in the least amount of time with the least amount of information). Purely random training converges more closely to the function one intends to learn, but takes substantially longer. The authors recommend first performing a neighbourhood sequential training to get the outputs near their final values, and then using random training to converge to the function being modeled.

When all training points have been run, the CMAC is ready for recall. Input values are supplied, which in turn excite a set of $C$ coarse regions. The final output is summed from the contributions of the excited regions according to

$$f = \mathbf{w}^T \mathbf{x} = \sum_{i=1}^{all \ hyperrects} w_i x_i$$

where $w_i$ is the weight associated with hyperrectangle $i$, and $x_i$ is 1 if the hyperrectangle covers the input point, and 0 otherwise. The final neural network architecture appears in Figure 2-7.

There are several clear advantages to using the CMAC ANN, as opposed to the ubiquitous backpropagation network; a very concise summary is provided by Miller et al. [1990]:

- CMAC effectively learns the mapping from a set of real variables to a real-valued output.
- Information is spread to topologically similar areas of the input domain, rather than globally throughout the network.
- Since fewer training iterations are required, large networks are trainable in a practical amount of time.
- The LMS equation is equivalent to a gradient descent performed on a quadratic surface, which has the desirable effect of finding a unique, global minimum [Miller, 1990].
- Recall is fast, requiring only a few simple arithmetic operations, performed on only those elements that cover the input.
- Even though the output is linear, CMAC can still learn nonlinear functions.
- Generalization allows CMAC to make reasonable output for inputs it has never encountered before.
- CMAC can be realized using fast, practical hardware. [Ker et al., 1997]

The primary disadvantages include:

- Poor training design can lead to local learning interference.
- The nonglobal nature of CMAC means that no emergent properties will arise out of training. In other words, it gives no insight into the problem it is being used to solve.

- Implementation may be memory intensive (but will still be less intense than a lookup table).

- If hash coding is used to lower memory requirements, unhandled collisions can add detrimental noise and lower the accuracy of recall.

Chiang and Lin [1996] prove that learning will converge to a limit cycle for a CMAC whose coarse quantizations are bounded in the input space. They also prove that with an infinite number of training points and a learning rate that asymptotically approaches zero, the error between the trained CMAC and the actual function is minimized in the least squares sense.

Kolcz and Allinson [1999] point out an interesting property of the CMAC, in that it can be conceptualized as a basis function network, and more particularly, as a General Memory Neural Network (GMNN). A coarsely quantized hyperrectangle is both fully and uniformly activated when input falls anywhere within its volume, and as such, can be thought of as a plateau basis function that is unity within its volume and zero everywhere else in the input space (and hence, satisfies only $C^0$ continuity). Both [Chiang and Lin, 1996] and [Kolcz and Allinson, 1999] point out that while it is used ubiquitously due to its ease of calculation, it is not the only basis function that can be used; in fact, any bounded function (such as a localized Gaussian) may serve as a basis, resulting in smoother, tighter local generalization. Because the CMAC falls under the GMNN framework, its mapping can be modeled as a basis function network. This model acts as an estimated or averaged version of the basic performance characteristics of the original network. They are more mathematically tractable, and hence can be used to analyze the behaviour and expected performance of the network.

2.3.2 The Feature CMAC

The Feature CMAC is so named because it borrows the coarse quantization scheme from the Albus CMAC. It is, however, a fundamentally different neural network as will be shown below. This summary reiterates the theory found in [Carusone and D'Eleuterio, 1998].
The FCMAC architecture can be found in Figure 2-8. Unlike the CMAC which can be allocated in memory before training begins, the FCMAC is built online as training takes place. The input layer is a two-dimensional grid, which is a direct one-to-one correspondence with the pixels in the image plane of the machine vision system. In the pose-estimation application training proceeds as follows: the robot moves to a nominal pose in the world coordinate system and the part is placed within its gripper. The robot then moves the part throughout its workspace at a variety of known poses, and at each point takes a digitized still image of the part in the workspace. The image is passed through a feature detection algorithm which returns the locations of feature pixels in the image plane. Unlike the CMAC which takes only one active input value (e.g., the operating point of a mechanical system), the FCMAC takes all the feature pixels at once, forming a constellation pattern in the input domain. Again, it should be pointed out that pose determination could be performed using this high resolution data; however, such processing would be less robust to missed features and would not generalize for poses it has never seen. Instead, the input domain is coarsely quantized into $C$ layers, exactly as is done in the CMAC. The coarse pixels formed in these offset layers are referred to as layer pattern neurons. They are binary neurons that become activated if one or more input neurons in their receptive field are activated. Again, note that there are $C$ layer patterns for a single input constellation, and that the overall system still has pixel resolution. If the pose contains a new layer pattern, a new neuron is added and connected to its constituent input neurons.$^3$ The active layer pattern neuron from each of the $C$ layers is linked to an image pattern neuron. One image pattern neuron is added for every pose in the training set, and has associated with it the known pose coordinates of the part. Learning continues, adding image pattern and layer pattern neurons as necessary, until all poses in the training set have been added to the network.

![Figure 2-8: The Feature CMAC neural network architecture.](image)

---

$^3$ It should be noted that in this thesis the network deals with only one type of feature at a time. Multiple differentiated features can be handled by simply adding another set of $C$ layers for each additional feature.
During recall, the part is placed within the trained workspace of the robot. An image is taken and processed, and the resulting features are passed to the input layer of the FCMAC. The excitation of the $j$-th layer pattern neuron in the $i$-th layer, $L_{ij}$, is simply the arithmetic mean of its activated constituent input neurons

$$L_{ij} = \frac{\sum_{k=1}^{c_i} w_{ijk}}{c_{ij}}$$

(4)

where $c_{ij}$ is the number of connected input neurons and $w_{ijk}$ are the outputs of said neurons. Likewise, the $j$-th image pattern neuron will have an activation, $I_j$, based on the mean of its connected layer pattern neurons

$$I_j = \frac{\sum_{i=1}^{C} L_i}{C}$$

(5)

where $L_i$ is the connected layer pattern from the $i$-th layer, and $C$ is the quantization factor. The image patterns with activations $I_j$ and known poses $P_j$, contribute to the final pose-estimation according to

$$\hat{P} = \frac{\sum_{j} I_j^2 P_j}{\sum_{j} I_j^2}$$

(6)

where $j$ runs over every image pattern neuron from the set of training poses. A threshold based on the most highly activated image pattern is applied so that only the most strongly activated patterns contribute to the final pose-estimation⁴. If $P_j$ falls below the threshold, its value is set to zero.

The advantages of the FCMAC approach to pose-estimation include:

- **Freedom from calibrated models:** No information is known about the part, other than its nominal grasp pose and its feature patterns in different poses. The camera-to-robot frame transformation, camera projection nonidealities and kinematic nonidealities of the manipulator are also unknown, but are automatically accounted for in the training procedure. Since the system needs no internal representations, it is ready to be trained as soon as the camera's position has been fixed and the manipulator can grasp the target part.

---

⁴ The threshold significantly decreases noise without compromising generalization. In this research the threshold is set to 85% of the maximum image pattern activation. The threshold may be changed at will without retraining the network.
• **No weight training:** The network's performance is based purely on the self-organized topology created during training. The major computational overhead is the image processing, not the training of connections as is the case with a backpropagation network. Additionally, training is complete in a single pass through the known poses.

• **Biological emulation:** The mammalian vision system builds from simple features to more complex representations in order to develop recognition [Oram and Perrett, 1994]. A similar construction takes place in the FCMAC, where simple features (such as corners) are joined into increasingly complex patterns in the higher layers of the network.

• **Good interpolation:** The information spreading accomplished by the CMAC-style coarse quantization allows for good generalization and reasonable estimates for poses the network did not encounter during training.

• **Robustness:** The FCMAC is fault tolerant during recall in that it can miss a few features and still accurately find the target part. Performance degrades gracefully with an increasing amount of missed features. The network is also robust with respect to lighting conditions, to an extent depending primarily on the choice of image processing algorithm. Background features are rejected if they occur in all of the training images, or if they occur during recall in an image region that did not contain a feature during training. Assuming erroneous features do not create an ambiguous view of the target part, they will not excite the neurons in the network to as great an extent as will the target part.

The disadvantages of the FCMAC include:

• **No extrapolation:** While the system interpolates very well within the trained workspace, no information is known outside of its boundaries and hence no extrapolations can be made.

• **Slightly decreased accuracy:** Most model-based methods are able to locate target parts to subpixel accuracy, whereas FCMAC estimates on the order of 1.5 pixels. However, the aforesaid implementations usually rely on heavy calibration and part-specific, one-of-a-kind implementations, and as such are very inflexible to changes in operating condition and parts. For the fixtureless assembly workcell, it is more important to have the freedom and flexibility provided by the FCMAC ANN than increased accuracy.

• **Sensitive to camera perturbation:** Once training is completed, the camera must remain rigidly fixed with respect to the robot's world coordinate frame. Any small misalignments will seriously degrade the accuracy of the system.
• **Resource overhead:** During training, the actual manipulator must be used to move the part through its workspace\(^5\). Training can take several hours (as much as 8 hours in the current setup), which are hours the workcell could have been used to manufacture products. However, considering the length of industrial manufacturing runs, training time is probably a relatively small expense.

The Feature CMAC ANN is not limited to robotic target pose-estimation. The identical architecture can be used to determine a satellite's orientation in inertial space based on star patterns. Each star acts as a point feature, and each image pattern is associated with the azimuth, elevation, and roll variables of the satellite.

---

\(^5\) Strictly speaking, the actual manipulator does not need to be used during training. However, poses generated by other means would not capture any of the nonidealities of the real manipulator, a significant advantage of the FCMAC ANN.
3 PREVIOUS RESEARCH

3.1 Experimental Setup

The experimental setup for research prior to this thesis is shown in Figure 3-1, with an accompanying block diagram in Figure 3-2.

![Figure 3-1: The experimental setup. A) Robotic manipulator. B) CCD camera. C) Robot controller hardware. D) Desktop PC.](image)

![Figure 3-2: Functional block diagram of the experimental setup.](image)

The robotic subsystem was manufactured by CRS Robotics in Burlington, Ontario. The manipulator is a 6 degree-of-freedom (DOF) A465 human scale manipulator, equipped with a standard servo gripper. A diagram of this robot, colloquially known as “Alanis”, is shown in Figure 3-3. Its motion is commanded by a C500 controller that contains all of the kinematics and servomotor controls necessary for motion co-ordination. Interface to the controller box is done by the RobComm 4.32k software which operates a serial link to the controller and provides a Dynamic Data Exchange (DDE) link that can be accessed by external applications. The coordinate frame of the experiment coincides with the world frame of the robot, as shown in Figure 3-4.
Figure 3-3: The CRS A465 6-DOF manipulator, "Alanis".  

Figure 3-4: The world coordinate frame.

Visual input for the system is provided by a single Hitachi KP-M1 black-and-white CCD camera, mounted at a fixed location relative to the robot's base. A BitFlow Raptor card grabs 640 x 480, 8-bit monochrome images from the camera, and provides them to external applications through its Raptor Software Development Kit (SDK).

The FCMAC application was written by Joe Carusoone, a research associate in the UTIAS Space Robotics Group. Programmed in Visual C++ 5.0, this application handled the DDE link to RobComm and the SDK calls to the Raptor board, as well as the feature detection algorithms and neural network code. The PC on which all of the software ran was a Pentium I, 166 MHz with 256M of RAM, running Windows NT 4.0. In this configuration, the PC was known by the name "Random".

Development of the code for this thesis began with the above experimental setup. After two years the computer subsystem was upgraded to a Pentium III, 800 MHz with 256M of RAM, and the Raptor board was replaced with a DataTranslation DT3131 capture board and its associated SDK. The epithet "Ariel" refers to the PC in this configuration. Development of the improved feature detection and added application functionality was written using Visual C++ 6.0.

3.2 Previous Task

The FCMAC architecture was originally developed outside of the concept of fixtureless assembly. As such, the experimental goal was to confirm the validity of the FCMAC theory. The following subsections describe the state of the FCMAC system prior to this thesis and the results of particular experiments run on it.
3.2.1 The Lego Target Part

The original sets of experiments dealt with the pose-estimation and grasping of a Lego piece, with dimensions 96 x 64 x 21 mm (see Figure 3-5). Lego was chosen partly because it allowed flexibility in the shape and colour of the part, was easily manipulated, and was robust to physical punishment in case of accidental mishandling by the manipulator. However the primary advantage to Lego stemmed from its geometric properties; the image of the part always contained very definite edges and corners, greatly facilitating feature detection.

The shape of the part was chosen specifically for its stability when placed by the manipulator. During every FCMAC training, if the part does not lie flush with the robot's workspace when the gripper releases, it will bounce a very small distance before coming to rest. Should this happen, the FCMAC will believe the part to be in a slightly different pose than it actually is. While this may be a very small error, it still reduces the overall accuracy of the trained network. The hourglass configuration of the part was found to result in less jogging on release.

3.2.2 Simple Feature Detection Algorithm

Since the Lego part could be completely described by hard edges and corners, a complex feature detection algorithm was unnecessary. In order to maximize the speed of performance of the image processing, a very simple feature-extraction code was written (hereafter referred to as the Simple algorithm or Simple detector). The Simple algorithm consists of two stages: edge extraction and corner extraction.

Figure 3-6 is a pictorial depiction of how the Simple detector works. During edge extraction a square window (typically 3 x 3 pixels) is passed over the image. If the difference between the maximum

![Figure 3-5: The original Lego target part.](image)

![Figure 3-6: The Simple detector. A) Edge detection based on local contrast. B) Corner detection based on finding two contiguous lines of edge pixels.](image)
and minimum intensities within the window exceeds a certain threshold, the centre pixel is marked as an edge. Corner extraction looks for the intersection of at least two edges. Another square window (typically 17 x 17 pixels) is applied to every edge pixel. The border of this window is checked for edges. If an edge is found, it is traced to ensure that a contiguous line of edge pixels connects it to the centre. The border continues to be traced until a second line (not 180° from the first) is encountered, in which case the centre pixel is marked as a corner, or until the border has been fully swept. Figure 3-7 shows the results of the Simple detector applied to a picture of the Lego part.

Figure 3-7: Results of Simple feature-extraction. A) The original image. B) The edge image. C) The corner image.

The Simple algorithm executed very quickly, and performed very well for the Lego target part under various lighting conditions.

3.2.3 Experimental Results

Table 3-1 and Figure 3-8 show the results of validation experiments using the flat Lego part (first published in [Carusone and D’Eleuterio, 1998]). In these experiments, the Feature CMAC had 48 layers, with an optimal 7 pixel tiling offset. Training points occurred every 10 mm in x and y, and every 10° in θ. RMS position errors were 1.0 mm in x and 1.7 mm in y, while that of orientation was 1.2°. These errors were sufficiently small to enable the robot to pick up the part after it had been placed in an arbitrary pose within the workspace.

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of Training Area spatial</td>
<td>102 x 102 mm</td>
</tr>
<tr>
<td>Resolution</td>
<td>10.2 mm</td>
</tr>
<tr>
<td>Angular Limits</td>
<td>[0°, 360°]</td>
</tr>
<tr>
<td>Angular Resolution</td>
<td>10°</td>
</tr>
<tr>
<td>RMS Errors: x</td>
<td>1.0 mm</td>
</tr>
<tr>
<td>y</td>
<td>1.7 mm</td>
</tr>
<tr>
<td>θ</td>
<td>1.2°</td>
</tr>
</tbody>
</table>

Table 3-1: Results of experiments using the flat Lego target part.
During these first experiments an interesting phenomenon of the system was discovered. The error histograms show a clear positive bias to the $y$ coordinate estimations. This is the result of the camera placement above the negative $y$-axis; poses further away (and hence smaller) look more closely alike (see Figure 3-9), and therefore tend to have higher activations in the network.

Investigations were made into part discrimination using the FCMAC architecture. The theory behind this extension is conceptually simple; a separate FCMAC network is trained within a common workspace for each target part. When presented with a target part, the features are passed through each of the trained FCMACs. The network with the highest activation should be the one corresponding to the target part.

Clearly, for radically different parts the recognition task was trivial. To push the limits of the system, the target parts were chosen to be three cartons of toothpaste made by three different
manufacturers. It can be seen in Figure 3-10 that all three cartons have the same physical dimensions, but the printed copy contains features that are distinct to each brand of toothpaste. No data has been published from these experiments, but it is the author's understanding that the correct cartons were recognized with limited success. No related experiments were performed for this thesis, since the fixtureless assembly workcell will not require recognition capabilities.

Figure 3-10: The three toothpaste cartons used for FCMAC recognition tests.
4 CURRENT RESEARCH

The experimental development described in Section 3 proved that the Feature CMAC theory could be effectively applied to a real robotic system. However, the choice of target part was contrived for ease of feature detection, grasping, and robustness to mishandling. In order to usefully apply the vision system to the problem of fixtureless assembly, the system must be improved to the level where it can accurately locate real-world industrial parts. Unlike Lego, which is comprised solely of sharp lines and corners, industrial parts can have virtually any amount of curvature in their appearance. Their metal surfaces are subject to specular reflections from light sources. As well, their shapes may lead them to bounce when released, causing increasing errors through the multiple pick-and-place operations required during training.

It was the goal of this thesis research to improve the existing experimental system to the point where it could locate and grasp a real industrial part based solely on visual information. A secondary objective was, naturally, to make the aforementioned pose-estimation as accurate as possible. The part used in these experiments is shown in Figure 4-1. It was chosen because it was the smallest component of the "strawman" task of the IRIS fixtureless assembly workcell, and as such could be manipulated without exceeding the limits of the existing human-scale manipulator.

Examining the objective lead to two clearly definable areas of improvement: the choice of feature detection algorithm, and the method of manipulation of the part. The Feature CMAC neural network architecture was not modified for this thesis.

4.1 Feature Detection

As has been mentioned in Section 2.3.2, the accuracy of the FCMAC network depends greatly on the number and quality of features detected. Given too few features or too precise features hampers the ability of the network to generalize in the input domain between training poses (Figure 4-2A). Conversely, given too many features
or too widely distributed features, the network loses sensitivity to small changes in the visual image (see Figure 4-2B). The FCMAC requires enough high-level feature clusters to describe the part accurately, with each cluster containing an appropriate amount of activated pixels to smooth the sharing of information.

The Simple edge and corner detector that performed well on the Lego part failed to accurately describe the industrial part, as can be seen in Figure 4-3. Small bends in otherwise straight lines contained too many features, and some curves with higher curvature were not noticed at all. Shadows obscured some edges, resulting in fewer corners. Clearly, a new form of feature detection was required to describe this part.

![Figure 4-3: Results of Simple detection on the metal part. A) The original image. B) The edge image. C) The corner image. Note the number of large clusters of features.](image)

After performing a search of the current literature (see Section 2.1), it was decided to implement the Single Unvalue Segment Assimilating Nucleus (SUSAN) algorithm developed by Smith and Brady [1997], and the Parametric Feature Manifold (PFM) system by Baker, Nayar, and Murase [1998].

### 4.1.1 SUSAN Algorithm

#### 4.1.1.1 Theory of the SUSAN Algorithm

The SUSAN method [Smith and Brady, 1997] is based on pixels of similar intensity in a local area (see Figure 4-4). A circular window is passed over an image. The region inside the window that has a similar intensity to that of the centre (or "nucleus") is called the Univalve Segment Assimilating Nucleus, or USAN. Areas of interest will correspond to a smaller ratio of USAN area to window area.
Figure 4-4: The SUSAN detector (from [Smith and Brady, 1997]). A) The circular detector window is passed over an image containing a dark rectangle on a white background. B) The USAN area is shown in white. C) An inverted graph of the USAN area peaks at regions of interest such as corners and edges.

The USAN area is found by applying a comparison function, \( c(r, r_0) \), within the domain of the window. A simple heavyside step filter can be used

\[
c(r, r_0) = \begin{cases} 
1 & \text{if } |I(r) - I(r_0)| \leq t \\
0 & \text{if } |I(r) - I(r_0)| > t 
\end{cases}
\]

(7)

where \( r \) is a point in the window, \( r_0 \) is the nucleus, \( I(r) \) is the image intensity at \( r \), and \( t \) is the comparison threshold. However, the authors recommend using the comparison function

\[
c(r, r_0) = \exp\left[-\left(\frac{I(r) - I(r_0)}{t}\right)^6\right]
\]

(8)

which performs better because of its smoothness near the thresholds (see Figure 4-5).

Figure 4-5: The comparison functions used in [Smith and Brady, 1997].
A) Heavyside step function. B) Smoother 6-th order exponential function.

The USAN area, \( n(r_0) \), is calculated by summing comparison contributions throughout the window

\[
n(r_0) = \int c(r, r_0) \, dA
\]

(9)
Edge detection is performed by selecting regions whose USAN area is less than the geometric threshold, $g$

$$R_{\text{edge}}(r_0) = \begin{cases} g - n(r_0) & \text{if } n(r_0) < g \\ 0 & \text{otherwise} \end{cases}$$

(10)

where $g$ is normally set between 0.5 and 0.75 of the maximum value of $n(r_0)$.

Corner detection follows the same procedure, but sets the geometric threshold between 0.25 to 0.5 of $n_{\text{max}}$, and also applies a centre-of-mass constraint

$$|r_0 - r_{\oplus}| > m$$

(11)

where $m$ is a threshold and the centre-of-mass $r_{\oplus}$ is calculated using

$$r_{\oplus}(r_0) = \frac{\int_A r c(r, r_0) \, dA}{n(r_0)}$$

(12)

A contiguity constraint specifies that a line drawn from the nucleus to the window edge in the direction of the centre-of-mass must lie entirely within the USAN.

Two advantages of the SUSAN algorithm are that it is computationally simple (and hence, performs quickly), and it can find "corners" that have noticeable curvature to them. Smith and Brady successfully applied the SUSAN method to noisy synthetic images with great success, and to real images with limited fruition [1997].

4.1.1.2 Implementation of the SUSAN algorithm

The image processing code for the FCMAC used a discretized version of the SUSAN theory. Only corner detection was used, since edges were inappropriate for training the neural network. Window radius, binning factor (explained below), corner threshold and centre-of-mass threshold could all be set by the user.
Using the radius parameter, a discrete window was calculated

\[
W(i_0, j_0) = \{(i, j) \mid |(i - i_0, j - j_0)| \leq r \}
\]  

(13)

where \(i\) and \(j\) are the image pixel coordinates and \(r\) is the window radius (see Figure 4-6). In images of the industrial part, a typical corner could be detected using windows with radii between 16 and 24 pixels. However, a 24-pixel radius translated to a window area of 1809 pixels, which resulted in slow image processing when passed over a 640 x 480 image. To speed up the code, a binning factor was introduced that samples every \(6\)-th pixel in the real image

\[
W(i_0, j_0) = \{ (i_0 + bk, j_0 + bl) \mid |(k, l)| \leq r \}
\]

(14)

where \(k\) and \(l\) are dummy pixel coordinates relative to the nucleus \((i_0, j_0)\). For example, a radius of 8 and a binning factor of 3 covered an effective window of 24 pixel radius, using only 197 pixels.

The discrete comparison function, expressed as integral values of intensity difference, \(\Delta I\),

\[
c(\Delta I) = \exp\left(-\left(\frac{\Delta I}{\sigma}\right)^6\right)
\]

(15)

was precalculated into a lookup table for speed of execution. The USAN area was found by summing the contributions from each pixel in the window

\[
n(r_0) = \sum_{\mathbf{r} \in W(r_0)} c[\Delta I(\mathbf{r}, r_0)]
\]

(16)

while the centre-of-mass was simultaneously found using

\[
\mathbf{r}_\oplus(r_0) = \frac{\sum_{\mathbf{r} \in W(r_0)} \mathbf{r} c[\Delta I(\mathbf{r}, r_0)]}{n(r_0)}
\]

(17)
If the area and centre-of-mass criteria

\[
\begin{align*}
\mu(r_0) &< \eta \cdot \eta_{\text{max}} \\
|r_0 - r_\Theta| &> m
\end{align*}
\]

were satisfied, USAN contiguity was checked by examining pixels that lie between the nucleus and the centre-of-mass (see Figure 4-7). If the USAN was contiguous in the direction of the centre-of-mass, the nucleus pixel \( r_0 = (i, j) \) was marked as a feature.

Figure 4-8 demonstrates a typical result of applying the SUSAN method to the industrial part.

Figure 4-8: The results of SUSAN detection on the metal part. A) The original image. B) The corner image.
4.1.2 Parametric Feature Manifolds

4.1.2.1 Theory of the PFM algorithm.

A Parametric Feature Manifold (or PFM) [Baker et al., 1998] is a mathematical construct that reduces the complexity of feature detection into a simple geometric problem. A feature class is represented by a parameterized intensity equation which is sampled to produce a manifold. Eigendecomposition is used to reduce the dimensionality of the manifold space. A sample taken from an image is called a feature if it lies sufficiently close to the manifold in its reduced order eigenspace.

Choosing the mathematical model of the feature one wishes to detect is the first step. Any feature class can be used, provided its intensity can be expressed as a continuous spatial function, \( f \), that depends on a set of parameters in the form

\[
I^c = I^c(x, y; q)
\]

where \( x \) and \( y \) are the spatial coordinates of the image, and \( q \) is a vector of brightness and geometric parameters. Typically, the dimension of \( q \) is between four and six, but any number of parameters may be specified. Both features used in this thesis had four parameters. The continuous function is then approximated by a discrete representation

\[
I^c \equiv I(i, j; q)
\]

where \( i \) and \( j \) are the pixel coordinates of the image. The functional description may be simple (e.g. a pure step edge sampled in orientation only) or complex (e.g. a nonlinear curve with sigmoidal blurring and sampling artifact). The features used in [Baker et al., 1998] are the step edge, ramp edge, line, corner, and circular disc (see Figure 4-9).

Figure 4-9: Feature functions used in [Baker et al., 1998].
Once the feature function is defined, it is sampled in a discrete window (as in Figure 4-6) for many combinations of parameter values. For notational simplicity, a sample can be recorded as an \( N \)-dimensional column vector of pixel intensities according to the scheme shown in Figure 4-10. The notation

\[
I(i, j; q) \rightarrow I(k; q)
\]

refers to the vector representation\(^6\), with \( k = 1, 2 \ldots N \). The brightness parameters can be removed by applying a mean/magnitude normalization, namely, making the average intensity zero and scaling so that the magnitude (in the Euclidean 2-norm sense) of the vector is unity. The original mean, \( \mu \), and magnitude, \( \nu \), are found using

\[
\mu(q) = \frac{1}{N} \sum_{k=1}^{N} I(k; q)
\]

\[
\nu^2(q) = \sum_{k=1}^{N} [I(k; q) - \mu(q)]^2
\]

and applied to normalize the sample using

\[
\tilde{I}(k; q) = \frac{1}{\nu(q)} [I(k; q) - \mu(q)]
\]

\(^6\)To reiterate, this mapping is purely for notational and computational convenience. \( I(i, j; q) \) sampled within a window and \( I(k; q) \) are both \( N \)-dimensional and contain the same values.
Normalization effectively reduces the dimensionality of \( q \) by two; its only remaining degrees of freedom are the geometric parameters.

One could very easily look for correlations between image samples and mathematical samples. Such a process would be computationally intense since a window with radius of 8 pixels creates a vector with \( N = 197 \) entries, all of which would have to be correlated. However, since the feature only has \( \sim 2-4 \) parameters, one would suspect that the 197-dimensional vector of theoretical pixel values contains some level of relatively unimportant information.

To reduce the dimensionality of the problem, and hence its computational intensity, the autocorrelation matrix \( R \) is calculated from the sampled vectors in the following manner

\[
R = E[(\vec{I} - E[\vec{I}])(\vec{I} - E[\vec{I}])^T]
\]

where \( E[\cdot] \) indicates the expected value operator. One then applies the Karhunen-Loéve (K-L) transform [Oja, 1983], which essentially solves the eigenvalue problem for the autocorrelation matrix

\[
Re = \lambda e
\]

The output of this transform is a set of \( N \) eigenvalues \( \lambda_i \) (where \( \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N \geq 0 \)) and their corresponding orthonormal eigenvectors \( e_i \). An eigenvalue is significant in that it quantifies how much feature information is encoded into its eigenvector. Conceptually, the eigenvectors are the basis vectors in a principle components representation of the feature. An eigenvector with a large eigenvalue is said to be more prominent. Figure 4-11B shows the most prominent eigenvectors for the corner feature, clearly demonstrating the order that arises out of the K-L analysis. Pairs of prominent eigenvectors appear as quadrature pairs of each other (i.e., rotated by ninety degrees about an axis).

One can now use a reduced subset of basis functions (in the form of the \( d \) most prominent eigenvectors) to encode the feature manifold without sacrificing a great deal of information. The K-L residue, \( R \), is defined as

\[
R(d) = \sum_{i=d+1}^{N} \lambda_i
\]

and is the measure of feature information that is rejected when one uses the \( d \) most prominent eigenvectors as the basis subset. In effect, the K-L transform reduces the dimensionality of the problem from \( N \) to \( d \). As one can see from Figure 4-11C, 90% of the feature information is encoded in the first 9 eigenvectors. Hence, by taking the first 9 eigenvectors as basis vectors, the problem's dimensionality has gone from 197 to 9 while discarding only 10% of the feature information.
Figure 4-11: The corner feature. A) Plan view. B) The 8 most prominent eigenvectors. C) The decay of the K-L residue. D) The manifold projected into the eigenspace of the first two eigenvectors. E) The projection using the first three eigenvectors.

The theoretically calculated samples are projected into the reduced order subspace formed by the prominent eigenvector set. Figure 4-11E illustrates the shape of the resulting manifold in the subspace formed by the first three basis vectors.

With the help of the reduced order manifold, feature detection becomes an easy problem (at least, conceptually): an image sample is normalized and projected into the subspace. If the projection lies sufficiently near to the manifold, the point in the image is labeled as a feature. In practice, the problem of finding the minimum distance from a novel point to the surface of a manifold is difficult and computationally intensive.

4.1.2.2 Implementation of the PFM Algorithm

Implementation of the PFM algorithm in this research differed significantly from that used by Baker et al. [1998]. Their goal was to find precise feature locations and to extract the best fit feature parameters. To accomplish this task, they included more higher-order parameters in their generation of feature manifolds; namely, they took into account camera sampling error and blurring from the lens. Also, they sampled their manifolds very densely, on the order of 50,000 points. By contrast, the goal of this thesis was to rapidly detect well localized clusters of features. In practice, it was found that the sampling and
blurring effects were well below the noise floor of a useful detection and added unnecessary complexity and detection time; they were therefore left out. Manifolds could be sampled as few as ~650 times without a significant loss in FCMAC performance.

A sampling window with a 4 pixel radius was used by Baker et al. Because they were looking for precise features, this window size was sufficient. However, the FCMAC input features had to be clustered. Typically, a 24-pixel radius window was large enough for the target image, since larger curves were better approximated by sharp corners. Using a smaller window, the curved geometry of the part was completely missed by sharp corner detectors. The binning method used for the SUSAN detector was also implemented in the PFM code (see Section 4.1.1).

Baker et al. used the step, ramp, line, corner, and circular disc features. However, only the latter two were applicable to the FCMAC. Manifold generation was performed by Matlab scripts, and detection was performed with Visual C++ source code.

Figure 4-11A shows a pictorial representation of the corner feature. This feature was sampled using

$$I(x, y; A, B, \theta, \theta_2) = A + B \cdot u[z(\theta - \theta_2/2)] \cdot u[z(180^0 + \theta + \theta_2/2)]$$ (27)

where

$$z(\theta) = y \cos \theta - x \sin \theta$$ (28)

and

$$u(t) = \begin{cases} 1 & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases}$$ (29)

where \(A\) and \(B\) were the intensities on either side of the edge (which are later removed by normalization), \(\theta\) was the orientation and \(\theta_2\) was the subtended angle\(^7\). The common parameter ranges are shown in Table 4-1. The eight most prominent eigenvectors are shown in Figure 4-11B. When projected into the subspace formed by the first three eigenvectors, the manifold appeared as in Figure 4-11E. As was mentioned before, the eigenvectors normally come in quadrature pairs. From Figures 4-11B and 4-11D, one can see that the first two eigenvectors coarsely coded the orientation parameter, \(\theta\), since the projection formed a circle in the plane of these eigenvectors.

\(^7\) This equation differs from the corner detector in [Nayar et al., 1998] for better performance of the customized coarse-to-fine algorithm (see Appendix D).
The circular disc results are shown in Figure 4-12. The feature function is given by

\[ I(x, y; A, B, \theta, r) = A + B \cdot u[r - p(x, y)] \]  

(30)

with

\[ p(x, y) = \sqrt{(x + r \sin \theta)^2 + (y - r \cos \theta)^2} \]  

(31)

where \( A \) and \( B \) were arbitrary intensities, \( \theta \) was the orientation, and \( r \) was the radius. Note that the localization parameter \( \rho \) was excluded for speed. The disc manifold contained 511 samples, from values given in Table 4-1.
Window radius, binning factor, magnitude rejection threshold and distance-to-manifold threshold could all be adjusted by the user. As has been mentioned earlier, the PFM algorithm is very computationally intensive, and as such is a very slow procedure when applied to every pixel in a 640 x 480 image. Rejection methods were first used to speed up the process. The Simple algorithm's edge detector was applied to the image, and only "edge" pixels were checked for features. Second, a magnitude threshold was applied during the normalization process. Any pixel whose patch did not exceed a specified magnitude (~400-700 grey levels) was rejected.

Finding the minimum distance from the novel point representing the image patch to the feature manifold was a challenging problem. The first solution was a linear (or "brute force") search, in which each sample in the manifold was checked, one at a time, until a subthreshold distance was returned, or the end of the manifold was found. As expected, this procedure performed poorly in terms of processing time, taking as much as 40 minutes (on Random) for a single pass over the image. Baker et al. [1998] suggest that a coarse-to-fine algorithm can be used to search the manifold, but give very few details on how such an algorithm can be implemented in code. A custom coarse-to-fine algorithm loosely based on a trinary tree approach was devised, the details of which can be found in Appendix D. With this search method, the processing time was reduced to 40 seconds per image on Random and 7.5 seconds on Ariel, missing only approximately 1% of features per image.

Typical results of applying the PFM corner detector to the metal part are shown in Figure 4-13.

<table>
<thead>
<tr>
<th>Manifold</th>
<th>Parameter</th>
<th>Min.</th>
<th>Max.</th>
<th>Resolution</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corner</td>
<td>Window radius</td>
<td>4 pixels</td>
<td>16 pixels</td>
<td>2 pixels</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>( \theta )</td>
<td>0°</td>
<td>360°</td>
<td>5°</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td>( \theta_2 )</td>
<td>30°</td>
<td>110°</td>
<td>10°</td>
<td>9</td>
</tr>
</tbody>
</table>

| Disc     | Window radius (R) | 6 pixels | 14 pixels | 2 pixels | -        |
|          | \( r \)           | 0.5 * R  | 2.0 * R   | 0.25 * R | 7        |
|          | \( \theta \)      | 0°        | 360°      | 5°        | 73       |

**Table 4-1: Common parameter ranges for manifold sampling.**
4.1.3 Parameter Selection for Image Processing

All three feature detection algorithms presented above (Simple, SUSAN, and PFM) require user-defined parameters to set their operating points. The choice of these parameters can greatly affect the performance of the algorithms for a given image (or in this case, for a given target part). It would be easy enough to hand-tune the parameters so that they perform optimally for the industrial part used in this thesis; however, that tuning is contrary to the motivating philosophy of fixtureless assembly, namely, the generalized applicability of the system to a wide variety of parts and processes. To combat the tuning problem, a simple ANN with unsupervised learning was implemented to find an acceptable set of parameters for a set of views of a target part. "Acceptable" in this case means good for the overall performance of the FCMAC neural network. Clearly, if one wanted precise feature localization and differentiation an entirely different set of parameters would be considered best.

In an ideal case, one would perform the full FCMAC training with each parameter set, and then test each to see which minimizes the RMS pose-estimation error. However, this procedure would be incredibly time and resource intensive, which is again contrary to the goal of flexibility in this project. An intermediate criterion was formulated, based on experience with the performance of the FCMAC. For each parameter set, a measurement of the "feature cost" of the detected features was performed. Feature cost is defined as

\[ C_L = \sqrt{\frac{1}{N} \sum_{i=1}^{N} |x_i - f_{0i}|^2} \]  

(32)

where \( x_i \) is the \( i \)-th feature, and \( f_{0i} \) is the cluster centre closest to it. In English, this is a measurement of how well the features were arranged into a finite set of definable clusters. Figure 4-14 demonstrates examples of high and low feature costs.
Clearly a method was required to find the centres of the clusters, without knowing the size or number of clusters \textit{a priori}; this was where the tuning ANN was required. The tuning ANN is an unsupervised topological map, loosely based on the Topology Representing Network (TRN) given in [Zeller et al., 1997], but without the interconnections between neurons. Basically, each neuron is a set of coordinates in the image space, which are then trained to find the best representation of the feature clusters in the image. Figure 4-15 shows the steps taken to find the clusters.

At the start of training, an image of the part in an arbitrary pose is acquired. Feature-extraction is performed using the method and parameter set to be investigated. \( M \) neurons are initialized randomly throughout the image space, with positions denoted \( \mathbf{w} \). Training then begins.

\[ \Delta \mathbf{w}_i = \gamma(\varepsilon, \lambda) \cdot (\mathbf{w}_i - \mathbf{x}) \]  
(33)

with
\[ \gamma(\varepsilon, \lambda) = \varepsilon(t) \cdot \exp \left( -\frac{i}{\lambda(t)} \right) \]  
(34)
where
\[ \epsilon(t) = \epsilon_i \left( \frac{\epsilon_f}{\epsilon_i} \right)^{\frac{t}{T}} \]  \hspace{1cm} (35)

and
\[ \lambda(t) = \lambda_i \left( \frac{\lambda_f}{\lambda_i} \right)^{\frac{t}{T}} \]  \hspace{1cm} (36)

where \( T \) is the total number of training intervals. The learning rate \( \gamma \) is a monotonically decreasing function which allows the neurons to find the gross positions rapidly at the start of training, and then fine tune the best locations at the end of training [Zeller et al., 1997].

At the end of training, any neuron that occupies the same vicinity as another is tagged as redundant and removed. Each feature then contributes its squared distance to the neuron closest to it
\[ D_i = D_i + |x - w_i|^2 \]  \hspace{1cm} (37)

where \( D_i \) is the running sum of squared distances for the \( i \)-th neuron. Once all features have been used, each neuron calculates the RMS distance to its closest pixels
\[ C_{li} = \sqrt{\frac{1}{n_i} D_i} \]  \hspace{1cm} (38)

where \( n_i \) is the number of features whose closest neuron is the \( i \)-th neuron. The final measurement of feature cost is determined by averaging the RMS distances from all neurons that have pixels associated with them
\[ C_L = \frac{1}{M} \sum_{i=1}^{M} C_{li} \]  \hspace{1cm} (39)

This formulation is functionally equivalent to equation (32), and is performed for ease of computation and code implementation. In this thesis, the following values were used: \( M = 35 \) neurons, \( T = 12500 \) steps, \( \epsilon_i = 0.5 \), \( \epsilon_f = 0.05 \), \( \lambda_i = 0.02M \), \( \lambda_f = 0.01 \). Figure 4-16 demonstrates a typical outcome of the tuning ANN for a single training set.

Results of feature cost analysis can be found in Section 5.2.1, with a detailed discussion in Section 6.2.1.
4.2 Manipulation Issues

In addition to improved feature-extraction, the other major challenge was the robotic manipulation of the real industrial part. Two subproblems had to be solved: the kinematic control of the manipulator and the search for a stable and reliable method of grasping the part.

4.2.1 Kinematic Control

Two key points must be borne in mind to understand the kinematic problems encountered by switching parts. First, in order to satisfy the smoothness and continuity constraints of the FCMAC ANN, the part must be manipulated relative to its geometric centre as expressed in the world frame of the robot. Figure 4-17 demonstrates the importance of this concept. For a small incremental change in orientation, the changes in feature pattern must be correspondingly incremental. If the part is rotated about an axis that does not pass through its geometric centre, particular features will move more than others. This causes inaccuracy when the feature pattern is compared with the part at a slightly different position and similar orientation. There is no clean interpolation between these poses. From another perspective, the closer the axis of rotation is to the geometric centre, the more well behaved the mapping from feature pattern to pose. If the axis of rotation lies outside of the geometric area of the part, the mapping function becomes pathological. For every pure rotation, there will always exist a rotation-and-shift that will appear more similar to the original feature pattern. As such, good interpolation becomes impossible.
Figure 4-17: The importance of proper rotation. A) Rough schematic of the part, showing its geometric center. B) Rotation about the geometric center. Arrows show the resulting displacement of corner features. C) Off-centre rotation. D) Rotation about a point outside the part. E) For any pure rotation about a point outside the part, there will always be a rotation and translation with smaller displacement of features. Hence, clean interpolation of all three coordinates becomes impossible.

The second key point has been mentioned in Section 3.1, namely, that high level set points are given to the robot's standard controller, which then performs all necessary kinematic conversions and low-level control of the joints. As such, the size of the trainable workspace is dependent on the reachable states of the manipulator. Each \((x,y)\) coordinate must be approachable from any of the allowable angles \(\theta\). Hence, the simpler the kinematics of picking up the part, the larger the trainable workspace.

In the original FCMAC experiments, manipulation of the part was trivial. The robot approached the part along the world \(z\)-axis, descending to a hard-coded point. The axis of rotation coincided with the geometric centre of the part, and as such, no tool transform was required. Changes in orientation were effected by a simple rotation of the robot's wrist roll axis. Figure 4-19 demonstrates this action clearly. Because of the simplicity of the kinematics involved with the part, the ANN could be trained with a very large workspace.

Figure 4-18: The tool flange frame of reference.

---

\[^{8}\] The tool transform specifies the position and orientation of the gripper fingers relative to the tool flange of the robot (see Figure 4-18). High level commands for motion about the end-effector (such as "Pitch" and "Yaw") are taken to be about the tool flange unless a tool transform is given.
In preparation for the eventual switch to the industrial part, the Lego part was modified to add a pitched approach angle of approximately 15° (see Figure 4-20). To satisfy the geometric centre constraint, the grasp point abutted the top crosspiece. References to the hard-coded nominal grasp point were removed from the RobComm control scripts and replaced with grasp height and approach angle variables that could be set from the VC++ code. As a result, changes in orientation became more complex for the controller, requiring a new, safe method of rotating the part to prevent it from contacting the workspace during long-angle yaw changes. Figure 4-21 shows the variety of manipulator configurations needed to place the part in a single spot in several orientations. As one can imagine, this greatly reduced the number of reachable poses, and as such shrunk the trainable workspace accordingly. Experiments on the pitched part were performed in a 51 x 61 mm rectangle, with the full range of orientation. Results are given in Section 5.1, in Table 5-1.
With a pitched approach angle, the manipulator must take on various configurations in order to effect a pure orientation change of the part.

With the advent of the new gripper (discussed in section 4.2.2), several steps had to be performed in order to ensure correct kinematic manipulation. First, the tool transform of the robot needed to be calibrated, which was performed in the following manner (illustrated in Figure 4-22): the target part was placed on a sheet of paper, and its projection on to the world $xy$-plane was traced out. The geometric centre of this projection was found and marked on the paper, with an accuracy of about $\pm 0.5$ mm. Next, the part and the paper were placed in the nominal grasp position in the workspace. The gripper's pose was adjusted to grasp the part correctly, with an accuracy of $\pm 0.3^\circ$ of pitch. This pose was used to define the $(0,0)$ reference point for the distance variables. Using the robot controller, the pose of the tool flange (i.e., the nominal point of end-effector measurement for the robot) was recorded. Pinning the calibration paper in place, the target part was
removed and the gripper was disconnected from the tool flange. The end effector was then moved incrementally until the tool flange rested on the paper and was aligned vertically with the geometric centre of the part. This procedure was accurate to within \( \pm 0.5 \) mm, with negligible error in pitch when compared to the previous step. Again, the controller recorded the tool flange pose. The difference between the two recorded poses was used as the tool transform. To reiterate, the tool transform was not from the tool flange to the centre of the gripper fingers, as it would be in a standard robotic manipulation; the transform was computed from the tool flange to the geometric centre of the part while being gripped (see Figure 4-22A). Overall, the tool transform was accurate to approximately \( \pm 0.7 \) mm and \( \pm 0.3^\circ \) of pitch.

Unfortunately, this was not the end of kinematic problems with the industrial part. It was discovered that the correct approach angle varied slightly with each activation and homing of the robot. This discrepancy was solved by adding a user-specifiable pitch adjustment and twist angle about the world \( x \)-axis\(^9\). The rotation matrix describing the new tool frame is given by

\[
C_{TP} = \begin{bmatrix}
\cos(P) & 0 & -\sin(P) \\
\sin(P)\sin(T) & \cos(T) & -\cos(P)\cos(T) \\
-\sin(P)\cos(T) & \sin(T) & \cos(P)\sin(T)
\end{bmatrix}
\]

(40)

where \( P \) is the pitch angle (typically \( 104^\circ \)) and \( T \) is the twist angle (typically \(-1^\circ\)). The robot controller operates on a 3-2-1 Euler angle system with the rotation matrix

\[
C_{321} = \begin{bmatrix}
c_1c_2 & s_1c_2 & -s_2 \\
-s_1c_3 + c_1s_2s_3 & c_1c_3 + s_1s_2s_3 & c_2s_3 \\
s_1s_3 + c_1s_2c_3 & -c_1s_3 + s_1s_2c_3 & c_2c_3
\end{bmatrix}
\]

(41)

where \( c_i = \cos(\theta_i) \) and \( s_i = \sin(\theta_i) \) [Hughes, 1986]. A small section of the VC++ code converted the two simultaneous rotations into the correct pitch, roll, and yaw angles for the tool frame by setting

---

\(^9\) A rotation about the \( z \)-axis of the tool frame would have been much easier to implement in code; however the world \( x \)-axis was chosen because it is the most easily visualized by a human operator (see figure 4-23).
\[ C_{321} = C_{TP} \]  

Typically, incremental adjustments to the nominal grasp pose were \( \pm 0.1^\circ \) in pitch, \( \pm 0.2^\circ \) in twist, and \( \pm 0.4 \text{ mm} \) in z-axis height.

It should be noted that the errors in tool transformation and nominal pose are accounted for during training. The part is still placed at known coordinates which are recoverable during recall. However, these errors do affect how close the manipulator comes to moving the part about its geometric centre; as a result, they can affect the interpolation qualities of the network during recall. However, these errors are justifiably acceptable for two reasons: first, they are small compared to the inherent inaccuracy of the FCMAC network, and second, the geometric centre calibration and its associated errors will occur in the real fixtureless assembly workcell, and hence their overall effects should be taken into consideration from the start.

### 4.2.2 Grasping the Part

Figure 4-24 shows a detailed view of the standard parallel-jaw servogripper. Note that the fingers were flat rectangles 19 x 26 mm across. The throw of the finger pads was between 45 and 77 mm. The original Lego target part was designed to be manipulated by this gripper, and was successful in the sense that grasping was robust to errors in z-axis orientation (up to \( 4^\circ \)), and position (\( \pm 3 \text{ mm} \) in \( x \) and \( \pm 22 \text{ mm} \) in \( y \)). However, the flat parallel jaw fingers were completely incapable of grasping the industrial target part, both in throw and in topological compatibility. Any grasping method implemented would have to be highly stable; the part had to be gripped several thousand times over the

\[ P = p + 90^\circ \]

\[ \theta_2 = \arcsin[\cos(p)] \]

\[ \theta_1 = \arccos \left[ -\frac{\sin(p)}{\cos(\theta_2)} \right] \]

\[ \theta_3 = \arcsin \left[ \frac{\sin(p) \cos(T)}{\cos(\theta_2)} \right] + 90^\circ \]

\footnote{While \( C_{TP} \) is derived relative to the world frame, it is conceptually simpler for a human operator to enter the pitch angle as \( p \), equivalent to \( P - 90^\circ \).}
course of a single training run, and had to be robust to pose error in order for successful grasping to occur during recall. As an added constraint, the part was not to be modified in any way, since that would both change its appearance to the vision system and would violate the principle of flexibility in the fixtureless assembly workcell. Clearly, a new gripper design had to be implemented in order to run experiments with the existing apparatus.

The solution to the gripper problem was solved by Dr. Gary Bone of McMaster University, the principal investigator into reconfigurable gripper design for the fixtureless assembly workcell. The L-brackets that formed the parallel jaws of the standard gripper were replaced by two small steel plates with three steel fingers screwed into them (see Figure 4-25A). The part was grasped as shown in Figure 4-25B, with one finger through the existing hole and the other two hooked on to the far edge. The use of three fingers on nonconvex curves automatically maneuvered the part to the same stable grasping configuration (see Figure 4-26) [Plut and Bone, 1996]. The cone at the tips of the fingers made them robust to insertion error, while the notch accounted for errors in the plane of grasping. To make the grasp more effective at self-aligning, the square edges of the part were smoothed to allow the edge to be more stably placed within the finger notches (see Figure 4-27). Note that this filing was done only at the grasp points, and had a negligible effect on the part's appearance in the training images.

Figure 4-25: A) The modified gripper. B) Grasp method.

Figure 4-26: Limited mobility grasping from [Plut and Bone, 1996]. A-D) As the gripper fingers spread apart, the part is drawn into alignment by the reaction forces.
Figure 4-27: The worn edges at the grasp points. This image was taken with the camera as close as possible to the part and with the maximum specular reflection. The worn regions were virtually unnoticeable in the training images.
5 EXPERIMENTS AND RESULTS

5.1 Pitched Lego Part

The experiment using the pitched Lego part had two goals. The first was to evaluate the efficacy of the variable-pitch robot control scripts, and the second was to directly compare the FCMAC's accuracy between the flat and pitched Lego parts. The same lighting conditions and training parameters as [Carusone and D'Eleuterio, 1998] were used; the only exception was the workspace size which was reduced due to kinematic constraints, as discussed in Section 4.2.1.

Table 5-1 and Figure 5-1 show the results of the pitched Lego experiment\(^\text{11}\). A discussion of the results can be found in Section 6.1.

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of Training Area</td>
<td>81 x 61 mm</td>
</tr>
<tr>
<td>Spatial Resolution</td>
<td>10.2 mm</td>
</tr>
<tr>
<td>Angular Limits</td>
<td>(0°, 360°)</td>
</tr>
<tr>
<td>Angular Resolution</td>
<td>10°</td>
</tr>
<tr>
<td>RMS Errors:</td>
<td></td>
</tr>
<tr>
<td>( x )</td>
<td>1.7 mm</td>
</tr>
<tr>
<td>( y )</td>
<td>1.5 mm</td>
</tr>
<tr>
<td>( \theta )</td>
<td>2.0°</td>
</tr>
</tbody>
</table>

Table 5-1: Results of experiments using the pitched Lego part.

\(^{11}\) These results were first published in [Langley and D'Eleuterio, 2000].
5.2 The Industrial Part

5.2.1 Feature Parameter Selection

The first step in experimentation on the Industrial part was to find useful sets of parameters for each of the feature detection algorithms. The unsupervised neural network discussed in Section 4.1.3 was implemented to find acceptable parameter sets. Five poses were used for feature cost evaluation, showing the part at approximately ± 85°, ± 45°, and 0° (see Figure 5-2).

Figure 5-2: Images for the feature cost analysis.

Neural network training was performed five times on each sample with different random initial conditions. The worst value for each pose was discarded to remove "outliers"; the remaining outcomes for all five poses were averaged to determine the overall value of feature cost for the current parameter set.

Table 5-2 lists the parameters used for each feature algorithm and the best combinations for feature cost. Figures 5-3 through 5-5 show the cost distributions broken down by parameter for each algorithm. A discussion of the tuning ANN analysis is given in Section 6.2.1.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameter</th>
<th>Unit</th>
<th>Min</th>
<th>Max</th>
<th>Resolution</th>
<th>Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>Edge</td>
<td>pixels</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Corner</td>
<td>pixels</td>
<td>10</td>
<td>25</td>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Min-max</td>
<td>grey levels</td>
<td>15</td>
<td>40</td>
<td>5</td>
<td>40</td>
</tr>
<tr>
<td>SUSAN</td>
<td>Radius</td>
<td>pixels</td>
<td>18</td>
<td>30</td>
<td>6</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>(effective)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Comparison</td>
<td>grey levels</td>
<td>30</td>
<td>70</td>
<td>10</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>Geometric</td>
<td>-</td>
<td>0.4</td>
<td>0.5</td>
<td>0.02</td>
<td>0.5</td>
</tr>
<tr>
<td>PFM</td>
<td>Radius</td>
<td>pixels</td>
<td>18</td>
<td>32</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>(effective)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Distance</td>
<td>-</td>
<td>0.3</td>
<td>0.7</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>Check range</td>
<td>nodes</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5-2: Feature cost analysis parameters and their best values.

Figure 5-3: Feature cost results by parameter for the Simple algorithm.
5.2.2 FCMAC Experiments

Once appropriate feature parameter sets were chosen, FCMAC training tests were performed. All training and testing images were generated once and saved to the desktop PC's hard drive. This procedure not only saved time (since the limiting factor was the speed of robot motion and communications to the controller box), but also guaranteed that all feature sets would be compared based on identical input data.
The training set consisted of 2299 images over the workspace defined in Table 5-3. One thousand testing images were generated at random poses within the trained workspace. Image data was gathered under reasonable lighting conditions with the camera in a similar pose to that of [Carusone and D’Eleuterio, 1998]. All experimental data were generated on the desktop PC Ariel (see Section 3.1).

<table>
<thead>
<tr>
<th>Coordinate</th>
<th>Min</th>
<th>Max</th>
<th>Resolution</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>-50.8 mm</td>
<td>50.8 mm</td>
<td>10.2 mm</td>
<td>11</td>
</tr>
<tr>
<td>y</td>
<td>-50.8 mm</td>
<td>50.8 mm</td>
<td>10.2 mm</td>
<td>11</td>
</tr>
<tr>
<td>θ</td>
<td>-90°</td>
<td>90°</td>
<td>10°</td>
<td>19</td>
</tr>
</tbody>
</table>

Table 5-3: Workspace for the industrial target part experiments.

The key performance measures from each experimental data point are the errors in \(x\), \(y\), and \(θ\). However, it is convenient for comparison purposes to have a single error value to describe the quality of a pose estimate. To this end, the composite error formula is introduced

\[
e(Δx, Δy, Δθ) = \sqrt{(Δx)^2 + (Δy)^2 + l_0 Δθ}
\]

(45)

where \(l_0\) is a characteristic length of the target part, and the linearization \(\sin(θ) ≈ θ\) for small \(θ\) is used. In the following experiments, the value of \(l_0\) was chosen to be 99 mm, the distance from the part’s geometric centre to the furthest grasp point. The resulting \(e\) value acts as a worst-case distance, in millimeters, between the target part’s actual and estimated grasp points. In the data analysis, \(E_1\) and \(E_2\) refer to the average composite error taken over an entire experimental run

\[
E_j = \frac{1}{N} \sum_{i=1}^{N} e_{ij}
\]

(46)

where \(j = 1\) denotes raw error, and \(j = 2\) denotes error after adjusting for bias offsets; \(N = 1000\) is the number of test poses per experiment.

The following variables were adjusted in each of the experiments:

- Feature type: Simple, SUSAN, or PFM.
- Feature parameters.
- Number of CMAC layers: 35, 48, 63, 80, or 99\textsuperscript{12}.

The following performance measures were generated for each training/testing run:

- Unadjusted RMS error in $x$, $y$, and $\theta$.
- Bias offsets in $x$, $y$, and $\theta$.
- RMS errors after adjusting for bias in $x$, $y$, and $\theta$.
- Training and testing time.
- FCMAC storage requirements.
- The composite error values, $E_1$ and $E_2$.

The experimental runs were performed in two stages. The first stage evaluated likely feature candidates based on the unsupervised ANN analysis and the author's intuition. Candidates that performed best in the first stage were used as a starting point for the second stage, in which individual variables were altered to examine their effect on FCMAC performance. In addition, the second stage gave an indication of the stability\textsuperscript{13} of the winning feature candidates.

Appendices A, B, and C contain the experimental results for the Simple, SUSAN, and PFM algorithms (respectively) in tabular and graphical form. Table 5-4 lists the best experimental runs for each algorithm. A full discussion can be found in Section 6.2.

\textsuperscript{12} The FCMAC code, which was generated before the start of this thesis, was hard-coded to obey the optimal tiling shift given in equation (1). As such, this thesis used integral values of tiling shift from 6 to 10, corresponding to quantization intervals 35, 48, 63, 80, and 99.

\textsuperscript{13} "Stability" refers to the detection algorithm's insensitivity to variations in its parameters. A stable algorithm will find approximately the same features in the same image, even though one or more of its parameters has been altered. This stability will also be reflected in the FCMAC error, since similar features will build a similar neural network.
<table>
<thead>
<tr>
<th>Run Name</th>
<th>RMS Error x (mm)</th>
<th>y (mm)</th>
<th>( \theta ) ( deg)</th>
<th>E1 x (mm)</th>
<th>y (mm)</th>
<th>( \theta ) ( deg)</th>
<th>E1 Adjusted Error x (mm)</th>
<th>y (mm)</th>
<th>( \theta ) ( deg)</th>
<th>E1</th>
<th>E2</th>
<th>E2 Adjusted Error x (mm)</th>
<th>y (mm)</th>
<th>( \theta ) ( deg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M0-63</td>
<td>1.56</td>
<td>1.94</td>
<td>1.82</td>
<td>4.48</td>
<td>-0.25</td>
<td>0.95</td>
<td>-0.04</td>
<td>1.54</td>
<td>1.69</td>
<td>1.82</td>
<td>4.30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUSAN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2-63</td>
<td>1.12</td>
<td>1.38</td>
<td>1.36</td>
<td>3.31</td>
<td>-0.10</td>
<td>0.66</td>
<td>-0.23</td>
<td>1.12</td>
<td>1.22</td>
<td>1.34</td>
<td>3.19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST-5</td>
<td>0.94</td>
<td>1.17</td>
<td>1.28</td>
<td>2.91</td>
<td>-0.10</td>
<td>0.71</td>
<td>-0.36</td>
<td>0.94</td>
<td>0.97</td>
<td>1.23</td>
<td>2.77</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PFM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P0-63</td>
<td>1.64</td>
<td>1.93</td>
<td>2.07</td>
<td>4.96</td>
<td>0.00</td>
<td>0.47</td>
<td>0.05</td>
<td>1.64</td>
<td>1.87</td>
<td>2.07</td>
<td>4.92</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC-140</td>
<td>1.59</td>
<td>1.76</td>
<td>2.04</td>
<td>4.74</td>
<td>-0.02</td>
<td>0.55</td>
<td>-0.02</td>
<td>1.59</td>
<td>1.68</td>
<td>2.04</td>
<td>4.68</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5-4: The best experimental results for the industrial target part.
6 DISCUSSION

6.1 The Pitched Lego Part

By comparing Tables 3-2 and 5-1, one can see that the FCMAC performed almost as well for the pitched Lego part as for the original part.

In 44 of the 708 poses of the pitched part, the FCMAC gave an estimation that was inverted in orientation (i.e. approximately ± 180° in error). These data were a characteristic of the target part's appearance rather than a performance error of the ANN and were therefore removed from calculation of the overall error. Orientation loss, in general, is caused when the target part has similar feature patterns in two or more dissimilar orientations. As Figure 6-1 demonstrates, there were 8 feature clusters found when the pitched Lego part was at +90°. However, a similar pattern of 8 clusters was contained within the feature image at -90°.

To demonstrate that this phenomenon was attributable to the part and not to the FCMAC, trials were run using the part shown in Figure 6-2. While the shape of the part remained the same, the feature image at +90° contained 4 feature clusters that were not present in that at -90°. In 100 test poses between -80° and -100°, the orientation estimation was not inverted a single time.

It is worth noting that the new pitched Lego part was still symmetrical but contained no ambiguous feature patterns. Most industrial parts will not have ambiguous views and so orientation loss should not occur in general. However, as will be shown in Section 6.2.4, extremely poor feature detection coupled with large quantization intervals can lead to orientation loss, and so should be kept in mind when the system is applied to the fixtureless assembly task.
6.2 The Industrial Part

6.2.1 Feature Parameter Selection

Clearly, the unsupervised ANN did not perform ideally. When run several times on the same image, it did not converge to a single feature cost value. Typically the output for a given pose had a standard deviation of 1.3 pixels which, over several trials, was sufficient for the purposes of determining the likely parameter sets for the target part.

The graphs shown in Figure 5-3 indicated that the only reasonable value for the Simple edge detection window was 1 pixel. Feature cost appeared to decrease with increasing values of corner radius and min-max threshold.

The feature cost function appeared insensitive to both the effective and actual radii for the SUSAN algorithm (see Figure 5-4). There was a slight trend toward lower cost with greater comparison values. Large centre-of-mass thresholds clearly decreased the feature cost.

PFM feature cost also appeared insensitive to window radius, as seen in Figure 5-5. However, there is a steep favour toward smaller values of the distance threshold.

Qualitatively, the feature cost function was not found to give the globally optimal parameter set for each feature algorithm. In particular, it tended to favour parameter sets with fewer clusters, which were not always beneficial for FCMAC detection. As will be shown later, many conclusions drawn from the feature cost analysis were shown to be invalid when applied to the FCMAC. For example, pose-estimation was more accurate with smaller values of the SUSAN comparison parameter. However, the unsupervised ANN results were useful in providing a starting point. Human intuition could then be used to find a parameter set that gave reasonable output for the part.

Investigation should be done into different mathematical definitions of feature cost. If a model can be found that gives a good overall value for feature clusters that lead to a well-trained FCMAC, the process of parameter selection could very well be automated. Even if a globally optimal feature cost measure cannot be formulated, a better one would at least decrease the number of trials needed to find a near-optimal parameter set.

6.2.2 Discussion of Simple Results

The second stage of experimental trials were not conducted with the Simple algorithm since early trials showed it to be unreliable. Many of the observations included in Appendix A did not show clear trends; coupled with the Simple algorithm's propensity toward orientation loss, it seemed futile to perform individual parameter analyses.
Figure A-1 shows the composite error for each feature set, averaged over trials with varying quantization intervals. Set M4, which had moderate values for the corner and min-max parameters, had the lowest average error.

The unreliability of the Simple algorithm is clearly demonstrated in Figure A-2. Almost every parameter set was subject to some amount of orientation loss, in some cases with as few as 63 CMAC layers. Sets M1 and M4 had reasonable performance in the 48 to 80 layer region. Figure A-3 displays the same results after adjusting for bias errors.

Figure A-4 demonstrates the excellent speed of the Simple algorithm. An FCMAC could be completely trained and tested within an hour using the M4 parameter set. Training time, as seen in Figure A-5, was independent of the number of layers; this observation was consistent for all three feature detection schemes. Testing time peaked at 63 layers for most parameter sets (Figure A-6).

Parameter set M4 with 63 layers had the best performance. Its error histograms are shown in Figure A-7, and its performance measures are listed in Table 5-4.

6.2.3 Discussion of SUSAN Results

Figure B-1 shows the composite error for each feature set, averaged over trials with several CMAC layers. Set S1 had the lowest average, followed by S2 and S7. These three sets had low values for the comparison parameter, with varying levels of sampling density.

A clear trend is shown in Figure B-2, indicating that a 63 layer FCMAC performed best for SUSAN detection of the industrial part. S2 had the lowest composite error, closely followed by S1. Figure B-3 shows the same trend after removal of the bias offset error from the pose estimates.

Looking at the processing times shown in Figure B-4, S1 and S2 had comparable times. S7, which had equivalent performance, took the greatest amount of time due to its large window area. The number of layers had no effect on training time, since most of the computational work during training was attributable to feature detection (Figure B-5). There was a slight dependence for recall due to the differing numbers of layer pattern neurons (Figure B-6). The fact that 63 layers took the longest time was outweighed by its better accuracy.

Based on the above observations, parameter set S2 with 63 layers was chosen as the overall winner of the first stage of experimental trials. Its error histograms are shown in Figure B-7, and its performance measures summarized in Table 5-4. S2's feature parameters were then varied independently for the second stage of trials.
The most surprising result of these experiments, as shown in Figure B-8, was the large region in which the SUSAN algorithm was insensitive to effective window radius. A stable region of excellent performance was found with effective radii in the range 24 to 34 pixels; the range 18 to 44 pixels also gave good results. These results indicate that the features that best describe the pose of the part have characteristic dimensions of approximately 30 pixels. As effective radius was lowered, performance destabilized due to the increasing number of false positive features, especially along the edges of the part. Too few features were found with detectors with large effective radii.

Figure B-9 demonstrates a clear quadratic rise in training time with respect to radius, which was expected since the number of operations per pixel was proportional to the area of the detector window. The effect was less clear with testing times due to the different FCMAC network topologies.

Results of varying the greyscale intensity comparison parameter, \( t \), are shown in Figure B-10. FCMAC performance was very stable in the 25-45 grey level range, but became much worse with more extreme values of \( t \). A larger threshold had the effect of increasing the USAN area past what would be rationally thought of as the boundary of the object. As such, less features were detected and network performance suffered. With a small comparison parameter, the USAN becomes too small to be practically useful. Obviously, the best value of the comparison parameter depends on the contrast settings of the camera. Considering that the experiments were performed with fairly typical lighting and contrast, the 25-45 grey level value will probably be appropriate for application to other industrial parts on a dark background.

Figure B-11 displays the results for variation of the threshold parameter, \( g \), which is the ratio of USAN area to window area. A stable region was observed for \( g \geq 0.45 \), which was consistent with Smith and Brady's [1997] claim that the optimal threshold was 0.5. Above 0.55, the centre-of-mass threshold \( m \) began to dominate, making the area threshold redundant. The FCMAC was very intolerant to smaller thresholds, since they quickly eliminated features and led to a poorly formed network.

The threshold \( m \), which specifies the minimum distance from the nucleus to the center-of-mass, was found, in practice, to be the most sensitive parameter. Typically, only a small range of values lead to an appropriate number of features. Figure B-12 shows that the optimal centre-of-mass threshold was 0.41 of the effective window radius. With a smaller threshold, more edge-like features were detected, leading to overgeneralization. Conversely, with a large value of \( m \), many viable corner features were rejected, resulting in the sharp rise in composite error from 0.44 to 0.47. Beyond 0.47, so few features were found that the FCMAC error became catastrophic.

Overall, the SUSAN algorithm performed very stably. Small changes in its parameters (with the exception of \( m \)) did not greatly affect the quality of FCMAC pose-estimation.
6.2.4 PFM Results

Experimental trials were performed using only the parametric corner detector. Several disc manifolds were generated and tested on images of the target part with complete lack of success. Figure 6-3 shows the grey-coded distance values for both the corner and disc features applied to the target part, which clearly demonstrate the fundamental weakness of the disc detector; unlike the corner detector, there is no parameter set that captures the weaker corners at the back of the part without including wide swaths along the edges of the part.

All of the first stage experimental trials were performed using a single corner manifold. Several manifolds, with various sampling densities and radii, were generated before one was decided upon. A strong trade-off exists between the increased performance associated with a higher sampling density, and the crucial speed criterion which determines the algorithm's applicability to the fixtureless assembly task. Not only is the size of the image sample vector proportional to the square of the window radius, but more eigenvalues are needed in order to capture the necessary amount of information. Figure 6-4 demonstrates this principle through the K-L residues for corner manifolds with varying radii. The 8-pixel radius, 657-sample manifold was the best manifold to capture enough information about the part without causing a drastic increase in computational intensity.

As one can see from Figure C-1, the PFM algorithm performed incredibly poorly with large numbers of layers (i.e., for higher quantization, C). The reason for this was that the relatively few numbers of features found led to a greater probability of orientation loss (see Section 6.1). Figure 6-5 demonstrates
this principle. Larger $C$ values indicate larger coarse pixels. Since each coarse pixel covers such a wide area, there will be a smaller number of activated coarse pixels for any given feature image. Layer pattern neurons will therefore have a smaller number of connected coarse pixel neurons. Additionally, the odds of exciting any given coarse pixel are much higher, leading to higher proportional activation of the layer pattern neurons. All of this leads to a greater likelihood of orientation loss.

![Figure 6-5: Coarse pixel activation for various quantization intervals. Features are shown in white, and activated coarse pixels are highlighted. From left to right, $C = 35$, $C = 64$, $C = 99$. In the latter, the sparseness of the coarse features is completely lost in the general activation.](image)

Five FCMACs from the first experimental stage demonstrated orientation loss. They were P1-80, P1-99, P3-80, P3-99, and P4-99. Figure 6-6 shows a scatter plot of the test pose $\theta$ estimations versus their actual $\theta$ values. For comparison, lines have been added to indicate $\pm 10^\circ$ of error. This figure clearly shows the areas where the FCMACs were prone to misjudgement of orientation. For example, P1-80 frequently mistook the object at $0^\circ$ to be at $+90^\circ$. Figure 6-7 shows the poses where the FCMACs lost orientation. Obviously, these candidates were rejected out-of-hand for use in the FFW, but for analytical curiosity the extraneous data points were removed and the performance values recalculated. Table C-2 lists the number of data points where orientation was lost, as well as the pose-estimation errors with and without the spurious points. In the remaining analysis, the data adjusted for orientation loss is used.

---

14 With the convention that A-B indicates an FCMAC trained with parameter set A and number of layers B.
Theta Estimate vs. Actual Theta

Figure 6-6: Scatter plot showing regions where orientation was lost.
Figure 6-7: Orientation loss of the metal part. Edges are shown in grey, features in white. The left column shows the actual position of the part, and the middle column shows the pose-estimation returned by the FCMAC. The right column shows a superposition of the two, with a square representing the size of a single coarse pixel. The FCMACs are, from top to bottom, P1-80, P1-99, P3-80, P3-99, and P4-99.
Figures C-3 and C-4 show set P3's instability with respect to number of layers, and the generally poor performance of P1. The remaining three parameter sets performed well, again with best accuracy occurring with 63 layers. Figures C-5 demonstrates a clear time/accuracy tradeoff, since the best performing parameter sets take the longest amount of time (note that times here are measured in hours, as opposed to minutes in the case of the SUSAN algorithm). Again, training time was virtually independent of the number of layers (Figure C-6). The variance observed with SUSAN recall is not evident because the PFM time accounted for a higher proportion of the total time spent during recall (Figure C-7).

Because it had the best accuracy and moderately good time performance, P0-63 was chosen as the basis parameter set for the second stage of experiment. Its performance measures are summarized in Table 5-4, and its error histograms are shown in Figure C-8.

The best value of effective radius was found to be 24 pixels, as shown in Figures C-9 and C-10. Figure C-9 was generated using the standard 8-pixel radius manifold with binning factors between 1 and 5. Several manifolds with radii between 6 and 12 were used with a binning factor of 3 to generate Figure C-10. More data points and higher radii were not tested due to the long processing times required for the PFM algorithm.

Figure C-11 shows the composite error with varying values of the magnitude threshold. A stable region existed from no threshold to 500 grey levels; thereafter, the performance degraded rapidly. A lower threshold led to some spurious features; however, these were few enough that the FCMAC did not become overgeneralized. Excessive rejection reduced the size of feature clusters, and removed some clusters altogether; the ANN no longer had enough information about the part to perform well. As one would expect, processing time decreased with greater magnitude threshold since the feature manifold had to be searched for fewer image locations; however, this gain did not nearly justify the drop in pose-estimation accuracy.

The stable region for the distance threshold was found to be between 0.45 and 0.7, as can be seen in Figure C-12. Below 0.45 there were too few features. Above 0.7 spurious features were found along the edges of the part, resulting in an overgeneralized network. Of the PFM parameters, the FCMAC performance seemed to be most insensitive to the distance threshold.

The effect of varying the check range for the coarse to fine algorithm is shown in Figure C-13. Below 60 nodes, the network performed poorly since the basic course-to-fine search missed a lot of features. Ranges greater than or equal to 60 nodes performed well, since they found much the same features as a linear search. One might expect that the performance should decrease monotonically with increasing check range, however that was not shown to be the case. It is quite possible that the feature patterns found with ranges of 60 and 140 nodes happened to describe the part's pose better than those with slightly greater check ranges.
A few trials were performed to examine the effect of manifold sampling density on FCMAC performance. As Figure C-14 shows, the change in composite error was very small, even for a manifold sampled five times as densely. Of all the trials, the 657-sample manifold (the one used for every other PFM trial) had the best performance.

In summary, the PFM algorithm was a more sensitive feature detector than SUSAN when applied to this target part. Once one reached limits of the stable region, performance tended to degenerate rapidly. Better performance might be achievable with a manifold that takes more nonidealities (such as blurring) into account; however, such a manifold would become prohibitively computationally intense for the FFAW application.

6.2.5 General Discussion

It is interesting to note that the storage size required for few layers was larger with SUSAN than with either Simple or PFM (see Figures A-8, B-13 and C-15). This was primarily because SUSAN found more features and therefore required more layer pattern neurons. However, at high quantization intervals, FCMACs trained with all three algorithms required approximately the same amount of storage (just over ~2000 kbytes). This was likely because as $C$ increased, all representations became so general as to appear identical no matter what method of feature detection was used.

Two properties of the industrial target part, when viewed under the experimental conditions, were found consistently throughout the data analysis. The first was that all three feature extractors performed best at scales near 24 pixels. The second property was that a quantization interval of 63 pixels yielded the most accurate pose-estimations. It seems clear that both of these values were characteristic of the part’s appearance, and would generally apply to any choice of feature detection applied to FCMAC pose-estimation.
7 CONCLUSIONS AND SUGGESTIONS

7.1 Conclusions

The direct conclusions of this thesis are as follows:

- The FCMAC ANN can repeatably locate a real, unmodified, industrial target object with sufficient accuracy for robot manipulation. As summarized in Table 7-1, the RMS errors of pose-estimation are within 1.0 mm of position and 1.2° of orientation. The average worst-case grasp point error is 2.8 mm for a 15.0 cm target part. The pose-estimation system uses standard, mass-produced technology without excessive monetary or time cost. A modification of the experimental setup will be readily applicable and integratable into a prototype fixtureless assembly workcell.

- The SUSAN algorithm [Smith and Brady, 1997] clearly outperformed both the Simple algorithm and the PFM algorithm [Baker et al., 1998] when applied to feature detection of the target part. FCMACs trained and tested with the SUSAN algorithm yielded more accurate pose-estimations. In addition, SUSAN was less computationally intense, and less sensitive to variation in its parameters. Although the best FCMAC trained with the Simple algorithm outperformed that of PFM, Simple was shown to be much more susceptible to orientation loss.

- The target part was found to have a characteristic feature length of approximately 24 pixels and an optimal quantization interval of 63, regardless of the choice of feature detection algorithm.

<table>
<thead>
<tr>
<th></th>
<th>Best Simple</th>
<th>Best SUSAN</th>
<th>Best PFM</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMS Errors:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x)</td>
<td>1.56 mm</td>
<td>0.94 mm</td>
<td>1.59 mm</td>
</tr>
<tr>
<td>(y)</td>
<td>1.94 mm</td>
<td>1.17 mm</td>
<td>1.76 mm</td>
</tr>
<tr>
<td>(\theta)</td>
<td>1.82°</td>
<td>1.28°</td>
<td>2.04°</td>
</tr>
<tr>
<td>Avg. Comp. Error E1</td>
<td>4.48 mm</td>
<td>2.91 mm</td>
<td>4.74 mm</td>
</tr>
<tr>
<td>Adjusted Errors:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x)</td>
<td>1.54 mm</td>
<td>0.94 mm</td>
<td>1.59 mm</td>
</tr>
<tr>
<td>(y)</td>
<td>1.69 mm</td>
<td>0.95 mm</td>
<td>1.68 mm</td>
</tr>
<tr>
<td>(\theta)</td>
<td>1.82°</td>
<td>1.23°</td>
<td>2.04°</td>
</tr>
<tr>
<td>Avg. Comp. Error E2</td>
<td>4.30 mm</td>
<td>2.76 mm</td>
<td>4.68 mm</td>
</tr>
</tbody>
</table>

Table 7-1: Numerical conclusions.
More important, however, the main contribution of this thesis has been to lay out the framework for implementation of the FCMAC ANN in a fixtureless assembly workcell. The following steps (shown in Figure 7-1) will need to be performed each time the fixtureless workcell is retrained for a new part or set of parts:

1) Several images of the part in representative poses are taken and stored. A computer operating offline processes the images, applying an appropriate performance measure for the number and quality of features. Parameters of the feature detection algorithm should be varied. (Section 4.1.3)

2) In the meantime, the appropriate tool transform to a frame located at the geometric centre of the part is calculated. (Section 4.2.1)

3) The robot then moves the part throughout its workspace, following a neighbourhood local training regime (Section 2.4). At each pose, an image is taken and saved to a file. (Section 5.2)

4) A set of random testing poses form within the trained workspace are also generated robotically and saved to a set of files. (Section 5.2)

5) Training and testing results are generated by an offline computer for several parameter sets that perform well in the initial analysis. Once a winning candidate is selected, its parameters are fine tuned by successively refining each independently. (Section 5.2)

6) The FCMAC that has the lowest RMS pose-estimation error (after adjusting for bias offsets) is now ready to be used for production in the fixtureless assembly workcell.
Figure 7-1: The necessary tasks for learning a new target part in the prototype fixtureless workcell.

7.2 Suggestions for Further Research

A few suggestions on feature detection readily present themselves from this work. First, more complicated feature models and more densely sampled manifolds should be generated for the PFM algorithm, in the hope that they will more accurately describe the target part location. These detectors will certainly be too slow for an industrial application, they may one day become viable as computing power increases. Other fast feature algorithms (such as [Harris and Stephens, 1988] and [Trajkovic and Hedley, 1998], which were not tested due to time constraints), should be tested to see if they perform as well or better than the SUSAN algorithm.

A more detailed look into the field of mathematical clustering may give rise to an improved feature cost function, which would give a better indication of how well an FCMAC would perform given a certain feature detector. This would cut down on the number of trial-and-error FCMACs that must be tested for a new target part, which in turn would decrease the total reprogramming time of the FFAW.

No doubt a host of new challenges will present themselves when the FCMAC vision system is integrated with the rest of the prototype FFAW. Most notable of these will be that the vision system must
deal with a wide range of part sizes and shapes (see Figure 1-3), as well as an alteration of viewing scale and angle from all extant experimental trials.

On the theoretical side, FCMAC code should be developed to incorporate multiple differentiable features into the network (see Section 2.4). The use of several features (such as corners and circular discs), or similar features with varying parameters (such as corners at different scales), would undoubtedly increase the range of objects to which the FCMAC could be applied. The network training may also be altered to include competition; in that way, a limited number of feature patterns could compete for the best representation of a given target part. The ANN could then be trained based on limited computer resources.
REFERENCES


Chiang C-T, Lin C-S, "CMAC with General Basis Functions", *Neural Networks*, 9(7): 1199-1211 (1996)


Mills JM, Personal conversation (2000)


APPENDIX A – SIMPLE ALGORITHM RESULTS

Table A-1: List of the Simple parameter sets.

<table>
<thead>
<tr>
<th>Set</th>
<th>Edge Parameter</th>
<th>Corner Parameter</th>
<th>Min-max Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>M0</td>
<td>1</td>
<td>25</td>
<td>40</td>
</tr>
<tr>
<td>M1</td>
<td>1</td>
<td>35</td>
<td>20</td>
</tr>
<tr>
<td>M2</td>
<td>1</td>
<td>16</td>
<td>40</td>
</tr>
<tr>
<td>M3</td>
<td>1</td>
<td>14</td>
<td>45</td>
</tr>
<tr>
<td>M4</td>
<td>1</td>
<td>20</td>
<td>30</td>
</tr>
</tbody>
</table>

Simple Error by Parameter Set

Figure A-1
Simple Error Value (E1) vs. Number of Layers

Figure A-2A

Simple Error Value (E1) vs. Number of Layers (Zoomed)

Figure A-2B
Simple Error Value (E2) vs. Number of Layers (Zoomed)

Figure A-3

Simple Processing Times by Parameter Set

Figure A-4
Simple Training Time vs. Number of Layers

Figure A-5

Simple Testing Time vs. Number of Layers

Figure A-6
Simple FCMAC File Size vs. Number of Layers

Figure A-7
Figure A-8: Error histograms for Simple FCMAC M4-63
APPENDIX B – SUSAN ALGORITHM RESULTS

<table>
<thead>
<tr>
<th>Set</th>
<th>Radius</th>
<th>Binning Factor</th>
<th>Comparison Parameter</th>
<th>Geometric Threshold</th>
<th>COM Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>8</td>
<td>3</td>
<td>50</td>
<td>0.38</td>
<td>0.46</td>
</tr>
<tr>
<td>S1</td>
<td>8</td>
<td>3</td>
<td>40</td>
<td>0.4</td>
<td>0.46</td>
</tr>
<tr>
<td>S2</td>
<td>8</td>
<td>2</td>
<td>40</td>
<td>0.4</td>
<td>0.44</td>
</tr>
<tr>
<td>S3</td>
<td>9</td>
<td>2</td>
<td>40</td>
<td>0.4</td>
<td>0.47</td>
</tr>
<tr>
<td>S4</td>
<td>9</td>
<td>2</td>
<td>60</td>
<td>0.4</td>
<td>0.44</td>
</tr>
<tr>
<td>S5</td>
<td>12</td>
<td>2</td>
<td>40</td>
<td>0.4</td>
<td>0.47</td>
</tr>
<tr>
<td>S6</td>
<td>12</td>
<td>2</td>
<td>60</td>
<td>0.4</td>
<td>0.46</td>
</tr>
<tr>
<td>S7</td>
<td>18</td>
<td>1</td>
<td>40</td>
<td>0.4</td>
<td>0.44</td>
</tr>
</tbody>
</table>

Table B-1: The SUSAN parameter sets.

SUSAN Error by Parameter Set

Figure B-1
SUSAN Error (E1) vs. Number of Layers

![Graph demonstrating SUSAN Error (E1) vs. Number of Layers.](image)

Figure B-2

SUSAN Error (E2) vs. Number of Layers

![Graph demonstrating SUSAN Error (E2) vs. Number of Layers.](image)

Figure B-3
SUSAN Processing Times by Parameter Set

Figure B-4

Training Time vs. Number of Layers

Figure B-5
Testing Time vs. Number of Layers

Figure B-6
Figure B.7: Error histograms for SUSAN FCMAC S2-63
SUSAN Error vs. Effective Radius

Figure B-8

SUSAN Processing Time vs. Radius

Figure B-9
SUSAN Error vs. Comparison Parameter

Figure B-10

SUSAN Error vs. Geometric Threshold

Figure B-11A
SUSAN Error vs. Geometric Threshold (Zoomed)

![Graph showing SUSAN Error vs. Geometric Threshold (Zoomed)]

Figure B-11B

SUSAN Error vs. Centre-of-Mass Threshold

![Graph showing SUSAN Error vs. Centre-of-Mass Threshold](image)

Figure B-12A
**Figure B-12B**

SUSAN Error vs. Centre-of-Mass Threshold (Zoomed)

**Figure B-13**

SUSAN FCMAC File Size vs. Number of Layers
APPENDIX C – PFM ALGORITHM RESULTS

<table>
<thead>
<tr>
<th>Set</th>
<th>Feature</th>
<th>Binning Factor</th>
<th>Magnitude Threshold</th>
<th>Distance Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>Corner</td>
<td>3</td>
<td>400</td>
<td>0.5</td>
</tr>
<tr>
<td>P1</td>
<td>Corner</td>
<td>3</td>
<td>700</td>
<td>0.5</td>
</tr>
<tr>
<td>P2</td>
<td>Corner</td>
<td>3</td>
<td>1</td>
<td>0.4</td>
</tr>
<tr>
<td>P3</td>
<td>Corner</td>
<td>2</td>
<td>700</td>
<td>0.6</td>
</tr>
<tr>
<td>P4</td>
<td>Corner</td>
<td>2</td>
<td>1</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table C-1: The PFM parameter sets.

<table>
<thead>
<tr>
<th>FCMAC</th>
<th>Avg. Error EL Before Removal</th>
<th>Number of Datas Points Removed</th>
<th>Avg. Error EL After Removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1-80</td>
<td>8.25</td>
<td>21</td>
<td>7.26</td>
</tr>
<tr>
<td>P1-99</td>
<td>11.77</td>
<td>34</td>
<td>7.81</td>
</tr>
<tr>
<td>P3-80</td>
<td>9.08</td>
<td>11</td>
<td>7.22</td>
</tr>
<tr>
<td>P3-99</td>
<td>20.48</td>
<td>70</td>
<td>8.90</td>
</tr>
<tr>
<td>P4-99</td>
<td>9.06</td>
<td>18</td>
<td>5.99</td>
</tr>
</tbody>
</table>

Table C-2: PFM FCMACs adjusted for orientation loss.
PFM Error vs. Number of Layers (Unadjusted)

Figure C-1: Error includes orientation loss data points.

PFM Error by Parameter Set

Figure C-2
PFM Error Value (E1) vs. Number of Layers

Figure C-3

PFM Error Value (E2) vs. Number of Layers

Figure C-4
PFM Processing Times by Parameter Set

Figure C-5

PFM Training Time vs. Number of Layers

Figure C-6
PFM Testing Time vs. Number of Layers

Figure C-7
Figure C-8: Error histograms for PFM FCMAC P0-63.
PFM Error vs. Effective Radius (by Binning)

Figure C-9

PFM Error vs. Effective Radius (by Manifold)

Figure C-10
PFM Error vs. Magnitude Threshold

![Graph showing PFM Error vs. Magnitude Threshold]

Figure C-11

PFM Error vs. Distance Threshold

![Graph showing PFM Error vs. Distance Threshold]

Figure C-12
PFM Error vs. Check Range

Figure C-13

PFM Error vs. Sampling Density

Figure C-14
PFM FCMAC File Size vs. Number of Layers

Figure C-15
APPENDIX D – PFM ALGORITHM SPEED IMPROVEMENTS

One pressing concern with the implementation of the FCMAC system in a real fixtureless assembly workcell is that of processing time. If image processing were to be done offline, computing time would not be a concern. However, the FCMAC system will be trained using real manufacturing hardware that will be costly to have immobilized while an image is being processed; this delay occurs not only during the required thousands of training poses, but also each and every time it attempts to locate a part.

As has been noted earlier, the Parametric Feature Manifold (PFM) image processing algorithm [Baker et al., 1998] is computationally intense. It involves the following stages:

- Mean/magnitude normalization of an image patch
- Projection into a reduced order eigenspace
- Finding the minimum distance from the novel point to the manifold

The first two stages, while obviously requiring several floating point operations, were easily minimized in terms of complexity. However, the distance finding stage was quite open ended and took the majority of the computational effort. It must also be borne in mind that the process had to repeat twice in order to find both light features on a dark background and dark features on a light background. Pixel rejection allowed the algorithm to be run fewer times for a single image; but any efforts to speed up the PFM algorithm itself had to focus on the distance finding algorithm.

D.1 Rejection

By inspection of a typical target part image (see Figure 4-1), it was obvious that applying the costly PFM algorithm to every pixel in the image would be a waste of time. To that end, the edge detection stage of the Simple algorithm was run over the image first. Since the Simple detector was a sloppy edge finder, it performed generally enough to capture any regions of interest that would contain PFM features. The PFM algorithm was then used only on those pixels marked as edges. Typically, this was between 10,000 and 23,000 pixels per image.

Next, the magnitude rejection scheme suggested in [Baker et al., 1998] was applied. Namely, any patch whose magnitude fell below a certain threshold (in the FCMAC case, between 400 and 700 grey levels) was deemed to be sufficiently uninteresting that it could not be a feature.

D.2 Coarse-to-Fine Algorithm

In preliminary trials of the PFM algorithm, a brute-force linear search of the manifold was performed; namely, every manifold point was checked, in order, against the novel point. Once a distance
below the threshold was found the algorithm would terminate; but considering that a great many of the points checked were not features, the general case was an exhaustive search of the entire manifold. Even using the most sparsely sampled manifolds \( N = 600 \) processing a single image still took 4 minutes on Random, the Pentium I computer.

Baker et al. [1998] recommended the use of a coarse-to-fine algorithm for searching the manifold space, but gave very few details on how such an algorithm would be implemented in code. The approach used in this work was to perform a loosely trisecting search using a recursive algorithm. Before getting into the details of the code, it is perhaps best to first take a look at the structure of a parametric feature manifold file.

As has been mentioned in Section 4.1.2 with the help of Figure 4-11, it was clear that the two most prominent eigenvectors encoded the gross orientation information about the corner sample. Figures 4-12 also demonstrates the same quality for the circular disc feature. This property was heavily exploited in the formulation of the coarse to fine algorithm. The feature manifold files were created with the orientation parameter \( \theta \) in the outermost loop, so traversing the file sequentially takes one monotonically around the circle formed in the plane of the first two eigenvectors. Figure 4-11 also shows the effect of the subtended angle parameter \( \theta_2 \), in that it spreads the manifold radially in the plane. The three most prominent eigenvectors will have the greatest effect on distance in the reduced order eigenspace, so it is conceptually worthwhile to abstract out the distance contributions that cannot be visualized in Figure 4-11.

Proceeding from these observations, the structure of the coarse-to-fine algorithm was chosen. As an example, imagine a manifold with two parameters \( a \) and \( b \), each of which is sampled \( n \) and \( m \) times, respectively. Looking at the manifold file spread out linearly, the parameters vary as follows (with \( m = 5 \)):

\[
\begin{array}{cccccccc}
  a_1 & a_1 & a_1 & a_1 & a_2 & a_2 & a_2 & a_2 \\
  b_1 & b_2 & b_3 & b_4 & b_2 & b_3 & b_4 & b_5 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
  a_3 & a_3 & a_3 & a_3 & a_3 & a_3 & a_3 & a_n \\
  b_1 & b_2 & b_3 & b_4 & b_5 & b_1 & b_2 & b_5 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
  b_4 & b_4 & b_4 & b_4 & b_4 & b_4 & b_4 & b_5 \\
\end{array}
\]

**Figure D-1:** Structure of a manifold file.

The following scheme was generated to segment the manifold file into successively coarser regions:
The zeroth level was the linear brute-force list of all sampled points. Level 1 was grouped by the number of subsamples of \( b \) (in the corner manifold, this was the subtended angle parameter \( \theta \)); in essence, this was a radially distributed group as seen in Figure 4-11. Subsequent layers formed a trinary tree until one reached the top, defined as the layer that contained between three and eight nodes. Note that this tree structure was determined automatically by the code, based purely on the number of \( b \) subsamples.

The recursive search algorithm handled the three cases of top level, zeroth level, and trinary tree separately. All nodes in the top level were searched, and the node with the lowest distance was taken as the root of the trinary tree. The trinary subtree was searched recursively based on the minimum distance node. The zeroth level was searched linearly. If at any time during the search the distance value was found to be below threshold, the search exited immediately. Figure D-3 displays the recursive search pictorially.
It should be emphasized that this coarse-to-fine search was not ideal, in the sense that it did not always return the minimum distance as found by the brute force search. However, it is a practical solution to the speed problem, based on \textit{a priori} knowledge of how the manifold had been constructed. A more rigorous coarse-to-fine algorithm was certainly possible, but would have required a great deal more programming. Since this was not a thesis in computer science, and since the FCMAC required only sloppy feature detection, the algorithm described above was chosen as a successful starting point.

Two heuristic improvements to the above search were implemented and tested (see Figure D-4). The first was to take the two closest nodes in the trinary search and recursively search both of their subtrees. The second was to take the closest zeroth level node found by the original scheme and linearly search the \( n \) nodes preceding and succeeding it in the file. These search styles were run on representative data, along with the plain coarse-to-fine algorithm, in order to determine which yielded the best output. A trade off was necessary between the criteria of minimal time and minimal distance error. The winner was chosen to be the subsequent linear search with a value of \( n = 80 \) nodes to either side.

![Figure D-4](image)

\textbf{Figure D-4:} A) The best 2-of-3 algorithm. B) The range checking algorithm. Note that both schemes can find the actual best node in cases where the plain trinary search tree would fail.

Again, it should be noted that the FCMAC did not require precise feature locations, and so using this algorithm and relaxing the distance threshold was an acceptable practice. The 80 node subsequent linear search did not produce any false positives, missed only a few features, and reduced the complexity by a factor of 3.6 on average. Translated into real time, a single image took 40 seconds to be processed on Random, and only 7.5 seconds on Ariel. The critical time element once again became the speed of robot motion and the serial communication link to the robot controller.