



# Kent Academic Repository

**Boyle, Timothy (2007) *Computer tools to assist the diagnosis of dyspraxia.*  
Doctor of Philosophy (PhD) thesis, University of Kent.**

## Downloaded from

<https://kar.kent.ac.uk/94227/> The University of Kent's Academic Repository KAR

## The version of record is available from

<https://doi.org/10.22024/UniKent/01.02.94227>

## This document version

UNSPECIFIED

## DOI for this version

## Licence for this version

CC BY-NC-ND (Attribution-NonCommercial-NoDerivatives)

## Additional information

This thesis has been digitised by EThOS, the British Library digitisation service, for purposes of preservation and dissemination. It was uploaded to KAR on 25 April 2022 in order to hold its content and record within University of Kent systems. It is available Open Access using a Creative Commons Attribution, Non-commercial, No Derivatives (<https://creativecommons.org/licenses/by-nc-nd/4.0/>) licence so that the thesis and its author, can benefit from opportunities for increased readership and citation. This was done in line with University of Kent policies (<https://www.kent.ac.uk/is/strategy/docs/Kent%20Open%20Access%20policy.pdf>). If you ...

## Versions of research works

### Versions of Record

If this version is the version of record, it is the same as the published version available on the publisher's web site. Cite as the published version.

### Author Accepted Manuscripts

If this document is identified as the Author Accepted Manuscript it is the version after peer review but before type setting, copy editing or publisher branding. Cite as Surname, Initial. (Year) 'Title of article'. To be published in **Title of Journal**, Volume and issue numbers [peer-reviewed accepted version]. Available at: DOI or URL (Accessed: date).

### Enquiries

If you have questions about this document contact [ResearchSupport@kent.ac.uk](mailto:ResearchSupport@kent.ac.uk). Please include the URL of the record in KAR. If you believe that your, or a third party's rights have been compromised through this document please see our [Take Down policy](https://www.kent.ac.uk/guides/kar-the-kent-academic-repository#policies) (available from <https://www.kent.ac.uk/guides/kar-the-kent-academic-repository#policies>).

COMPUTER TOOLS TO ASSIST THE  
DIAGNOSIS OF DYSPRAXIA

A THESIS SUBMITTED TO THE UNIVERSITY  
OF KENT  
FOR THE DEGREE OF DOCTOR  
OF PHILOSOPHY  
IN ELECTRONIC ENGINEERING

By  
Timothy Boyle  
August, 2007

# Abstract

Dyspraxia is a condition which relates to an impairment in the development of motor coordination, and which has been estimated to affect around 6% of children. It is a highly debilitating condition, seriously undermining both academic and social development. It is, however, a condition which can be difficult to diagnose, and one which can be reflected in different aspects of behaviour.

This thesis discusses an approach to the assessment of Dyspraxia in which computer tools can be used to support the detailed monitoring and analysis of a child's behaviour during standardised testing procedures. Current methods of diagnosis require specially trained clinical staff and can be very time consuming. We have taken one of the standard assessment tests, the Beery Developmental Test of Visual-Motor Integration (the VMI Test), and investigated improvements which can be achieved using computer software. The proposed improvements relate both to supporting more objective and automated support for test analysis, but also to the provision of options to extract characteristics of behaviour which are traditionally difficult to extract using conventional procedures. We have developed software to provide support to a clinician scoring the VMI. We have produced a generalised assessment system for any figure copying task. This has been adapted to score the VMI test. Novel approaches to diagnosing Dyspraxia have also been discussed. On-line data capture provides an assortment of measures which are unavailable to a scoring system based solely on a static image.

Overall, the work reported describes and evaluates a package of tools which can be used to support diagnostic procedures in routine clinical practice.

# Contents

<b>Abstract</b> .....	<b>i</b>
<b>Contents</b> .....	<b>ii</b>
<b>List of Tables</b> .....	<b>v</b>
<b>List of Figures</b> .....	<b>vii</b>
<b>Acknowledgements</b> .....	<b>xiii</b>
<b>1 Introduction and Background</b> .....	<b>1</b>
1.1 Introduction .....	1
1.2 Aims and Objectives .....	3
1.3 Developmental Dyspraxia.....	3
1.4 The VMI Test and Computer Interaction.....	10
1.5 Thesis Outline .....	11
1.6 Summary .....	13
<b>2 Testing Environment and Infrastructure</b> .....	<b>14</b>
2.1 Introduction.....	14
2.2 Developmental Test of Visual-Motor Integration (VMI) .....	14
2.3 Graphics Tablet .....	20
2.4 Modified VMI Test.....	21
2.5 Data Collection.....	23
2.6 Tools Used .....	25
2.7 Software Design .....	27
2.8 Summary .....	28
<b>3 Assisted Scorer</b> .....	<b>29</b>
3.1 Introduction.....	29
3.2 State Manager.....	34

3.3	Database View Page.....	37
3.4	Patient Summary Page .....	40
3.5	Single Shape Display .....	42
3.6	Importing Data .....	48
3.7	File Output .....	53
3.8	Conclusion.....	56
<b>4</b>	<b>General Scorer.....</b>	<b>57</b>
4.1	Introduction.....	57
4.2	POI Identification.....	63
4.3	Template Matching .....	80
4.4	Absolute Rule Framework .....	91
4.5	Conclusions.....	97
<b>5</b>	<b>Scoring VMI using General Scorer .....</b>	<b>98</b>
5.1	Introduction .....	98
5.2	Definition of Terms.....	99
5.3	Line Calculation Algorithms.....	100
5.4	VMI rules .....	104
5.5	Templates .....	128
5.6	Results.....	132
5.7	Conclusions.....	136
<b>6</b>	<b>Dynamic Movement Analysis.....</b>	<b>137</b>
6.1	Introduction.....	137
6.2	Movement Modelling.....	138
6.3	Rapid Aimed Movements .....	139
6.4	Filtering curves.....	144
6.5	Finding Appropriate Curves.....	144
6.6	Parameter Estimation .....	148
6.7	Multiple Rapid Aimed Movements.....	153
6.8	Modelling Shape 15 curves.....	158
6.9	Modelling Shape 15 Arrowheads.....	159
6.10	Conclusion.....	162
<b>7</b>	<b>Dynamic Shape Analysis.....</b>	<b>163</b>
7.1	Introduction.....	163
7.2	Counting rapid-aimed movements .....	164

7.3	Software Analysis .....	166
7.4	Straight line sections .....	168
7.5	Cognitive Analysis .....	171
7.6	Conclusions .....	177
<b>8</b>	<b>Conclusion.....</b>	<b>178</b>
8.1	Overall Research Conclusion.....	178
8.2	Suggestions for Future Research.....	181
8.3	Summary .....	182
	<b>Bibliography .....</b>	<b>183</b>

# List of Tables

Table 1 - State groupings .....	36
Table 2 - Data capture parameters .....	53
Table 3 - Template matching classes .....	86
Table 4 - Template .....	89
Table 5 – Child’s Drawing.....	89
Table 6 - Comparison of search methods.....	90
Table 7 - Set of Absolute Rules for VMI.....	105
Table 8 - Control Subjects Success Rate.....	133
Table 9 - Known Dyspraxic Success Rates.....	134
Table 10 – RAM parameters from arrowheads of shape 15. ....	161
Table 11 - Average number of RAMs.....	165
Table 12 - Drawing strategies for VMI Square.....	171
Table 13 - Number of pen strokes for VMI Square .....	172
Table 14 - Time taken to draw each side (ms).....	172
Table 15 - Percentage of time spent on each side .....	173
Table 16 - Standard deviation of velocity .....	173
Table 17 - Mean velocity of pen (cm/s).....	173
Table 18 - Drawing strategies for VMI Open Square and Circle .....	174
Table 19 - Number of pen strokes for VMI Open Square and Circle .....	174
Table 20 - Time taken to draw each side (ms).....	174
Table 21 - Percentage of time spent on each side .....	174
Table 22 - Standard deviation of velocity .....	175
Table 23 - Mean velocity of pen (cm/s).....	175
Table 24 - Time taken to draw each side (ms).....	176
Table 25 - Percentage of time spent on each side .....	176

Table 26 - Standard deviation of velocity .....	176
Table 27 - Mean velocity of pen (cm/s) .....	176



# List of Figures

Figure 2.1 - Vertical Line.....	15
Figure 2.2 – Three-Dimensional Star.....	15
Figure 2.3 – Complete set of shapes used in VMI.....	16
Figure 2.4 - Vertical Line.....	17
Figure 2.5 - Oblique Cross.....	18
Figure 2.6 - Six Circle Triangle .....	19
Figure 2.7 - Three-Dimensional Star .....	19
Figure 2.8 - Wacom Intuos 2 Tablet .....	20
Figure 2.9 - VMI test booklet.....	22
Figure 3.1 List of imported patients.....	31
Figure 3.2 Importing data from modified VMI.....	32
Figure 3.3 Patient Summary.....	32
Figure 3.4 Single Shape with measurements .....	33
Figure 3.5 VMI Assist State Diagram.....	35
Figure 3.6 Database View Page .....	37
Figure 3.7 Loading the database .....	39
Figure 3.8 Debug Screen.....	40
Figure 3.9 Shape Summary Page .....	41
Figure 3.10 Single Form Display .....	42
Figure 3.11 Zoom into drawing .....	44
Figure 3.12 measurement tools .....	46
Figure 3.13 Dialog box for dividing subjects drawing .....	49
Figure 3.14 Drag Drop .....	50
Figure 3.15 Class Diagram of subject data storage.....	51

Figure 3.16 Database File Format .....	54
Figure 3.17 - Colour coded summary of shape scoring .....	55
Figure 4.1 - VMI Shape 20 .....	58
Figure 4.2 - Scoring system interface .....	60
Figure 4.3 - Test Harness .....	61
Figure 4.4 - Scoring Results.....	62
Figure 4.5 - POI Identification Class diagram .....	63
Figure 4.6 - Initial State .....	65
Figure 4.7 - Zero Velocity Detection Enabled .....	68
Figure 4.8 - Zero Velocity Detection Disabled.....	68
Figure 4.9 – Initial Interest Points.....	69
Figure 4.10 - Beginning Clustering.....	71
Figure 4.11 - Clustering Complete.....	72
Figure 4.12 - Connections.....	73
Figure 4.13 - Only two points found on circle .....	74
Figure 4.14 - Extra point added.....	74
Figure 4.15 - Only one point found on circle.....	75
Figure 4.16 - Two extra points added .....	75
Figure 4.17 - POI Identification Complete .....	76
Figure 4.18 - Line with a false corner .....	77
Figure 4.19 - First erroneous cluster detected.....	77
Figure 4.20 - Second erroneous cluster missed.....	78
Figure 4.21 - Second erroneous cluster detected .....	78
Figure 4.22 - All two connection clusters removed .....	79
Figure 4.23 - All one and two connection clusters removed.....	79
Figure 4.24 - Matching to template.....	80
Figure 4.25 - Numbered Points .....	81
Figure 4.26 - Creating Template .....	82
Figure 4.27 - Template match found.....	83
Figure 4.28 - Class Diagram for Template Matching Stage .....	86
Figure 4.29 – Simple Example Shape.....	87
Figure 4.30 - Example Child’s Drawing .....	88
Figure 4.31 – Exhaustive Search Tree .....	88
Figure 4.32 – Initial Optimization.....	89

Figure 4.33 – Final Optimization .....	90
Figure 4.34 - Script for shape 11 of the VMI.....	92
Figure 4.35 - AbsAng Display .....	93
Figure 4.36 - CompLeg Display.....	93
Figure 4.37 - Invalid Command.....	93
Figure 4.38- VMI shape 17 .....	94
Figure 4.39 - Higher Rules.....	95
Figure 5.1 – Inner Line.....	101
Figure 5.2 – Inner Line with False Corner .....	102
Figure 5.3 - VMI shape 11 .....	102
Figure 5.4 - Finding Three Cluster Line .....	103
Figure 5.5 - Invalid template showing syntax.....	104
Figure 5.6 – Command syntax for AbsAng .....	106
Figure 5.7 - Form 11 .....	107
Figure 5.8 - AbsAng Calculation .....	107
Figure 5.9 – Command Syntax for RelAng .....	107
Figure 5.10 - Form 20 .....	108
Figure 5.11 – RelAng Calculation .....	108
Figure 5.12 – VMI shape 7 .....	110
Figure 5.13 - Command Syntax for LineAng .....	110
Figure 5.14 – Computing LineAng .....	110
Figure 5.15 - Command Syntax for AllLineAng .....	111
Figure 5.16 - Command Syntax for ClusDist.....	112
Figure 5.17 - Form 20 .....	112
Figure 5.18 - ClusProp Calculation.....	112
Figure 5.19 - VMI shape 17 .....	113
Figure 5.20 - ClusDist applied to shape 17.....	113
Figure 5.21 - Command Syntax for ClusProp.....	113
Figure 5.22 – VMI Shape 19.....	114
Figure 5.23 - Break down of ClusPop command.....	114
Figure 5.24 - ClusProp Calculation.....	115
Figure 5.25 - Command Syntax for CompLeg.....	115
Figure 5.26 – VMI shape 11 .....	116
Figure 5.27 - Form 11 with extension.....	116

Figure 5.28 - Command Syntax for MinLegLength .....	117
Figure 5.29 - Command Syntax for TwoInt.....	117
Figure 5.30 - Shape 11 with extension.....	118
Figure 5.31 - Form 14 showing intersection gap .....	119
Figure 5.32 - Command Syntax for ThreeInt.....	119
Figure 5.33 - VMI Shape 24 .....	120
Figure 5.34 - Least Squares Reference Circle (LSC).....	121
Figure 5.35 - Minimum Zone Circle (MZC).....	121
Figure 5.36 - Minimum Circumscribed Circle (MCC).....	122
Figure 5.37 - Maximum Inscribed Circle (MIC) .....	122
Figure 5.38 - Command syntax for Circularity .....	123
Figure 5.39 - VMI Shape 13 .....	123
Figure 5.40 - MIC and MCC circles .....	123
Figure 5.41 - Command Syntax for SideInt .....	124
Figure 5.42 - Form 17 showing SideInt .....	124
Figure 5.43 - Form 21 showing SideInt .....	124
Figure 5.44 - Form 20 .....	125
Figure 5.45 - Close up of Cluster 2 showing overlap .....	125
Figure 5.46 - Close up of Cluster 5 showing underlap .....	125
Figure 5.47 - Command Syntax for CornLine .....	126
Figure 5.48 - Flow diagram of CornLine algorithm .....	126
Figure 5.49 - Command Syntax for MustPass .....	127
Figure 5.50 - VMI shape 20 .....	128
Figure 5.51 - General Information .....	128
Figure 5.52 - VMI Criteria .....	129
Figure 5.53 - POI Identifier.....	129
Figure 5.54 - Shape 20 Template .....	130
Figure 5.55 - Template Matching.....	130
Figure 5.56 - Absolute Rule Section.....	131
Figure 5.57 – Overall Success Rates.....	135
Figure 6.1 - Modelled Velocity Curve. ....	140
Figure 6.2 - Synergy of Agonist and Antagonist Systems .....	141
Figure 6.3 - Modelled Velocity Curve from combined Agonist and Antagonist systems .....	142

Figure 6.4 - Movement Engram .....	143
Figure 6.5 - Drawing with a low speed/accuracy .....	143
Figure 6.6 - Drawing with a high speed/accuracy .....	143
Figure 6.7 - Creating the Matlab code .....	145
Figure 6.8 - Unfiltered Curve .....	146
Figure 6.9 - Butterworth filtered curve .....	146
Figure 6.10 - Overlaid Curves .....	146
Figure 6.11 - Shape 15 from the VMI test .....	147
Figure 6.12 - Parameter Estimation .....	149
Figure 6.13 - Batch Processing .....	149
Figure 6.14 - Well modelled Control .....	151
Figure 6.15 - Well modelled Patient .....	151
Figure 6.16 - Poorly modelled Control .....	152
Figure 6.17 - Poorly modelled Patient .....	152
Figure 6.18 - Curve with two peaks .....	153
Figure 6.19 - Modelling section 1 .....	154
Figure 6.20 - Modelling section 2 .....	154
Figure 6.21 - Initial Full Model .....	155
Figure 6.22 - Remodelling Section 1 .....	156
Figure 6.23 - Modelled Complete Curve .....	156
Figure 6.24 - Modelling Flow Diagram .....	157
Figure 6.25 - Comparing U1 .....	158
Figure 6.26 - Comparing U2 .....	159
Figure 6.27 - Velocity curve plots for arrowheads of shape 15 .....	160
Figure 7.1 - Sample Shape .....	164
Figure 7.2 - Interpretation 1 .....	164
Figure 7.3 - Interpretation 2 .....	164
Figure 7.4 - Interpretation 3 .....	164
Figure 7.5 - Measurements from the velocity curve .....	166
Figure 7.6 - Max Velocity against Max Proportion .....	167
Figure 7.7 – Vertical Line .....	168
Figure 7.8 – Horizontal Line .....	168
Figure 7.9 – Vertical-Horizontal Cross .....	168
Figure 7.10 – Directional Arrows .....	168

Figure 7.11 - 10 year olds Vertical Line ..... 169  
Figure 7.12 - 10 year olds Horizontal Line ..... 169  
Figure 7.13 - 6 year olds Vertical Line ..... 169  
Figure 7.14 - 6 year olds Horizontal Line ..... 169

# Acknowledgements

I would like to express my sincere thanks to my supervisor Prof. M. C. Fairhurst for his continual support and guidance throughout the whole duration of this work.

I would also like to thank Dr Farzin Deravi for helping me to make the difficult decision to return to University. Without whom this thesis would not exist.

My thanks also to the many friends and colleagues I have made during my time in Kent.

Finally, I would like to thank my wife for her continual support and encouragement.

# Chapter 1

## Introduction and Background

### **1.1 Introduction**

Dyspraxia is a condition which has been estimated to affect around 6% of children. It is an impairment in the development of motor coordination. The disability is known by many other names, the most frequently adopted of which are Developmental Coordination Disorder (DCD) and more commonly Clumsy Child Syndrome; this is quite a fitting name as the disorder affects fine motor control, and thus a child who is affected by the condition will often appear to be clumsy in his movements. Because Dyspraxia can be difficult to diagnose it is frequently not spotted at an early stage, hence Dyspraxic children can fall behind at school and are often accused of being misbehaved or disruptive. This can lead to them losing interest in their studies and feeling miserable in class, which in turn often leads to low self-esteem and a whole range of behavioural problems. Therefore it is important to catch the condition early.

Many forms of testing may be considered when investigating the sort of perceptual motor functioning which generally underlies the condition of Dyspraxia. These include:



- Tests of general motor ability, such as the Movement Assessment Battery for Children (Henderson and Sugden 1992) or the Bruininks-Osteretsky Test for Motor Proficiency (Bruininks 1978).
- Assessment of the child's Visual-Perceptual skills with the Test of Visual-Perceptual Skills (Gardiner 1989) and the Beery Developmental Test of Visual-Motor Integration (Beery 1997).
- Assessment of cognitive and language skills, using tests such as the Wechsler Intelligence Scale for Children (Wechsler 1991), or the Clinical Evaluation of Language Fundamentals (Semel et al. 1987).

The requirement for specially trained clinical staff, and the time taken to administer these tests, can place severe demands on available resources. The evaluation of tests can also be very subjective and inherently unreliable.

Figure copying tasks, which form the fundamental basis of tests such as the Beery Developmental Test of Visual-Motor Integration (VMI), lend themselves readily to computer assisted intervention. Recording pen movements, during the test performance, can be carried out in a fashion that does not hinder the test administration. The data recorded can then be assessed using a computer system. This thesis describes an investigation into ways that such an arrangement can be utilised to both improve the objectivity of test assessment and reduce the involvement by specialist clinicians.

The collection of drawing data from non-Dyspraxic children (essentially for use as control data) has been performed in collaboration with Blean Primary School (Kent, UK). A set of data from Dyspraxic children was available from a previous study, and this had been collected from the Mary Sheridan Child Development Centre (Canterbury, UK).

## **1.2 Aims and Objectives**

The objective of this research was to investigate ways in which computer software could be used to assist the diagnosis of Dyspraxia. The Beery Developmental Test of Visual-Motor Integration (VMI) was used as a starting point. The VMI test is used by paediatricians as a means of monitoring and assessing perceptual development difficulties in children. The primary aims of this work included the development of software to assist a clinician in assessing the VMI test, and software to perform the assessment automatically. These initial aims expanded throughout the development of this work to include an investigation into novel methods of diagnosing Dyspraxia using measurements which would not be available without the intervention of a computer system.

## **1.3 Developmental Dyspraxia**

Mandich and Polotajko (2003) describe Dyspraxia as a marked impairment in the development of motor coordination that significantly interferes with academic achievement or activities of daily living.

Gubbay (1975) described Dyspraxic children as clumsy children who display impaired performance of skilled movement.

Dyspraxia is a condition which often goes undiagnosed. Cousins and Smyth (2003) report that undiagnosed Dyspraxic children are often accused of misbehaviour and of being disruptive. This can lead to a range of associated psychosocial difficulties. Johnston et al (2002) state that an alarming number of secondary characteristics have also been identified, including:

- Problems with self-concept
- Poorer than expected educational achievement
- Emotional and behavioural difficulties.
- Low self-esteem.

The Diagnostic and Statistical Manual of Mental Disorders (DSM-IV), published by the American Psychiatric Association, is the handbook used most often in diagnosing mental disorders in the United States and internationally. Four criteria for the diagnosis of Dyspraxia are described in DSM-IV, specifically:

- A. Performance in daily activities that require motor coordination is substantially below that expected given the person's chronological age and measured intelligence. This may be manifested by marked delays in achieving motor milestones (e.g., walking, crawling, sitting), dropping things, "clumsiness," poor performance in sports, or poor handwriting.
- B. The disturbance in criterion A significantly interferes with academic achievement or activities of daily living.
- C. The disturbance is not due to a general medical condition (e.g., cerebral palsy, hemiplegia, or muscular dystrophy) and does not meet criteria for a Pervasive Developmental Disorder.
- D. If Mental Retardation is present, the motor difficulties are in excess of those usually associated with it.

### **1.3.1 Prevalence**

The majority of estimates give a prevalence of around 6% of the child population. However, the estimates range between 5% and 15%. Hamilton (2002) states that for children between the ages of 5 and 11 the estimate with the most scientific basis is 6.4%.

It has been widely reported that Dyspraxia affects more males than females (Henderson et al. 1992, Shoemaker and Kalverboer 1994, Miller et al. 2001). Kadesjo and Gillberg (1999) show male-female ratios ranging between 4:1 and 7:1.

The prevalence of Dyspraxia is also affected by other medical conditions. Ayyash and Preece (2003) report that 51% of extremely low birth weight children (<1500g) are

Dyspraxic. Smits-Engelsman et al (2003) indicate that a least 50% of children with a learning disability are identified with concomitant Dyspraxia.

### 1.3.2 Name

Dyspraxia has been known by many names since it was first identified as a condition in the 1870's. According to Dewey (1995) different professions have tended to use different terminology to describe the motor problems demonstrated by children with Dyspraxia. This has lead to a divergence of definitions and assessment criteria.

Some of the names used for this condition include:

- Dyspraxia
- Developmental Dyspraxia
- Agnosia
- Developmental Coordination Disorder (DCD)
- Developmental Awkwardness
- Sensorimotor Dysfunction
- Cerebellar Deficits
- Minimal Brain Damage (MBD)
- Congenital Maladroitness
- Motor Weakness
- Psychomotor Syndrome
- Physically Awkward
- Clumsy Child Syndrome

Following the DCD-V conference, Mandich and Polatajko (2003) suggest that the term "Developmental Coordination Disorder (DCD)" should be the term of choice and serve as a key word in all peer-reviewed publications. It was recognised at this conference that DCD is probably not the most accurate descriptor of the disorder. However this term, having gained acceptance, was enjoying broad use and has helped to unify the field.

We have chosen to use the term Dyspraxia in this thesis, as it both represents the term most often used in related work, and provides a greater degree of unification with other research that has been carried out within the Research Group.

### **1.3.3 Initial Referral**

Cousins and Smyth (2003) state that poor handwriting is one of the major reasons for the clinical referral of children with Dyspraxia. It is also not uncommon for an adult to be designated as Dyspraxic. Students may seek diagnosis because their movement difficulties are interfering with their academic progress and they may seek a statement of special needs.

Ayyash and Preece (2003) state that there has been an exponential increase in referrals requesting a service for this condition. This is due to increased awareness and acceptance of Dyspraxia.

A study performed by Cantell et al (2003) shows the usefulness of early screening as a part of a health examination of 5 year-old children. Coleman et al (2001) suggest that professional psychologists should be aware of the possibility of movement problems when conducting other tests. This can provide an early warning sign which could alert the Psychologist to refer the child on for movement assessment.

### **1.3.4 Subtypes**

There is increasing evidence in the research literature to suggest that there are different types of developmental disorders which are all called Dyspraxia (Cermak 1985, Cermak et al. 1980, Conrad et al. 1983, Dewey and Kaplan 1994). Dewey (1995) states that Dyspraxia is not a unitary disorder and different subtypes of the condition exist. Macnab, Miller and Polatajko (2001) take this further by suggesting that the most commonly noted feature of Dyspraxia is its heterogeneity.

Some children are generally impaired across tasks, but others show deficits in subsets of the tasks used. Cousins and Smyth (2003) show that difficulties may occur only in fine motor tasks, only in gross motor tasks, or in all motor tasks.

There have been several studies investigating sub-groups of the disorder. The most commonly cited studies are those by Dewey and Kaplan (1994), Hoare (1994) and Miyahara (1994).

### **1.3.5 Comorbidities**

Cousins and Smyth (2003) state that the coexistence of attentional and motor disorders is the norm. Visser (2003) notes that comorbidity is 'the rule rather than the exception'.

Some disorders frequently co-occur with symptoms of Dyspraxia, namely:

- Attention-deficit hyperactivity disorder (ADHD / ADD)
- Dyslexia
- Specific language impairment (SLI)

Kaplan et al (1998) cite growing evidence that it is the nature of the disorders themselves which explains the large degree of overlap between conditions. Many methodologies have been employed to look at brain structure and function in children with various types of developmental disabilities: brain scans, EEG, event-related potentials, and autopsy studies. Currently, there does not seem to be a one-to-one correspondence between biology and disorder.

A single underlying developmental condition may express itself in a wide variety of behavioural symptoms, depending upon the timing, location, and severity of the disruption in brain growth and development. Consequently, there is an increased focus on the development of theories that identify the common cause of these symptoms (Gillberg 1998, Gillberg and Kadesjo 1998, Kaplan et al. 1997, Kaplan et al. 1998).

### **1.3.6 Etiology**

There is no consensus on the underlying causes of Dyspraxia.

Kaplan et al. (1998) consider symptoms of Dyspraxia, Dyslexia and ADHD to be a reflection of the same underlying brain deficit, labelled 'atypical brain development' or ABD. The specific pattern of deficits is considered to depend on the extent and the location of the neurological abnormality.

It has been suggested in some studies that Dyspraxia may be the result of a conceptual impairment within the knowledge of gesture (Denckla and Roeltgen 1992, Ayres 1985).

Visser (2003) suggested that Dyspraxic children suffer from an automatization deficit. A fully automatized skill is one which can be performed with little or no conscious effort.

Piek et al. (2004) show that Dyspraxic children display greater motor difficulties in tasks which require executive functions. Executive functioning is the mental capacity to control and apply mental skills. Executive functions may include:

- The ability to sustain or flexibly redirect attention.
- The inhibition of inappropriate behavioural or emotional responses.
- The planning of strategies for future behaviour.
- The initiation and execution of these strategies.
- The ability to flexibly switch among problem-solving strategies.

The faster movements produced by Dyspraxic children, according to Dewey (1995), may be indicative of a difference in higher level functioning. However, Smits-Engelsman et al (2001) state that a quicker drawing strategy could be used to compensate for a lower level deficit. Increasing movement speed produces a phasic stiffness which can help filter out noise from motor commands (Van Gemmert and Van Galen 1997).

### **1.3.7 Treatment**

Ayyash and Preece (2003) describe various approaches for the treatment of Dyspraxia that have been proposed.

Process-orientated treatment focuses on the training of a specific sensory function. This method assumes that the benefits of such training will be transferred to a broad range of motor skills.

Perceptual motor training provides the child with a wide experience of sensory and motor tasks. General improvement in motor abilities is anticipated as a consequence of enhanced sensory and motor task experience. This is currently the most widely used method of treatment.

Task Orientated Intervention employs specific tasks in an attempt to improve corresponding skills. It focuses on direct teaching of the task to be learnt.

Ayyash and Preece (2003) go on to show that majority of studies are either poorly designed or show little evidence that one approach is better than another. They recommend a pragmatic approach using task-orientated intervention and suggest group-based work to increase self-esteem and family functioning.

### **1.3.8 Prognosis**

While some children seem to outgrow their motor problems, either with or without intervention, many others continue to show poor motor skills throughout adolescence and into adulthood (Geuze and Borger 1993). Adults who began with very similar childhood motor impairments might have very different outcomes. Cousins and Smyth (2003) speculate that this may depend on diagnostic, practice, motivational, educational and more general demographical factors.

Based on mounting empirical evidence Mandich et al (2002) suggests that Dyspraxia is a life-long disability that may continue to interfere with academic, social, and vocational development. Hay, Hawes and Faught (2004) also state that children are unlikely to grow out of Dyspraxia.

In contrast to this, Cousins and Smyth (2003) show that adults may have been able to develop compensatory mechanisms for dealing with their coordination impairments in



simple motor tasks. Dewey (1995) claims that Dyspraxic children can attain a high degree of skill in specific activities that they practised.

Cantell et al (2003) suggest that some young people catch up with peers during adolescence. One possible explanation for this is that as the body grows the brain recalibrates its neural representation to match. This adjustment produces a neural representation that is much more effective than before. Other explanations attribute the performance improvements to hormonal changes due to the onset of puberty.

### ***1.4 The VMI Test and Computer Interaction***

Computer technology is influencing testing procedures by providing new methods of administering, scoring and interpreting perceptual development tests (Anastasi 1968). A comprehensive review of automated psychological testing is presented in the work performed by Thompson and Wilson (1982).

The majority of tests which measure motor performance require the use of specialist equipment and trained staff to administer the testing methods. Testing procedures include tasks such as target practice, standing on one leg, stringing beads and various floor exercises.

Of all of the tests administered during the diagnosis of Dyspraxia, the Beery VMI test lends itself most suitably to automation because of its straightforward figure copying pen and paper format. Although some subjective judgements are required the majority of decisions rely on quantifiable shape measurements.

Smith (1987) describes many methods which have been used to analyse line drawings and convert image data into more meaningful geometrical information. Volans (1982) and Beaumont (1982) document the system requirements, advantages and disadvantages of interactive testing.

Further relevant work will be reviewed in the appropriate chapters.

## **1.5 Thesis Outline**

This thesis contains eight chapters, covering the following aspects of the work.

### **1.5.1 Chapter 1**

Chapter 1 provides an overview of the thesis. An introduction to the condition of Dyspraxia is provided.

### **1.5.2 Chapter 2**

The principal testing method adopted in this work, the Beery Developmental Test for Visual-Motor Integration (VMI), is introduced along with a description of how this testing procedure has been implemented within a framework designed to facilitate on-line data capture. The set of drawings used during our investigation is also discussed, including a data collection exercise performed at a local primary school. We also describe the set of software tools which were used during this work.

### **1.5.3 Chapter 3**

Chapter 3 suggests improvements to the scoring process for the VMI which can be achieved using on-line data capture. A software implementation of this improved process is presented.

### **1.5.4 Chapter 4**

Chapter 4 describes a system of using on-line data to provide automatic assessment of hand drawn shapes against a set of predetermined criteria. The 'general scoring' system presented here provides a framework which can be adapted to generate scoring decisions for a range of figure copying tasks including, but not restricted to, the VMI test. This system has been implemented as a complete software package.

### **1.5.5 Chapter 5**

Chapter 5 details how the general scoring software, described in chapter 4, has been extended to assess shapes according to criteria formally specified in the VMI manual. Scoring templates have been designed for each of the 27 shapes. A comparison has been performed between a manual assessment of the shapes and the software assessment.

### **1.5.6 Chapter 6**

Chapter 6 looks at using the dynamic movement data we gathered using real-time data capture, and describes how this data can be used to develop novel assessment methods for the diagnosis of Dyspraxia. We have attempted to recreate children's drawing strategies using a mathematical model. We investigate how the parameters of a mathematical model could be used to help the process of diagnosis.

### **1.5.7 Chapter 7**

Chapter 7 follows on from Chapter 6 looking a novel ways to use real-time data to assess the movement characteristics of a child. In this chapter we also look at drawing strategy.

### **1.5.8 Chapter 8**

Chapter 8 presents a conclusion of the whole of the work performed and documented within this thesis. Areas which should be considered for further research are also suggested.

## **1.6 Summary**

This thesis investigates how an on-line data capture approach can be used to improve the evaluation of motor functioning in Dyspraxic children. A computerised support tool has been developed to assist a clinician in providing assessments using the Beery developmental test for Visual-Motor Integration (VMI). This is taken to the next level when we propose a system to automatically perform assessment of shapes against a set of criteria. This investigation has been extended to develop novel techniques which provide an initial step towards developing a new class of testing procedures.

The following chapters describe the proposed systems and present software implementations that have been developed.

# Chapter 2

## Testing Environment and Infrastructure

### ***2.1 Introduction***

This chapter introduces the Beery Developmental Test of Visual-Motor Integration (VMI). This is the movement assessment test which has been used as a basis for this work. We describe modifications to the administration of this test which enable on-line data to be captured without affecting the outcome of the test.

Data has been collected from a local primary school to augment a set of data produced by a local Paediatric Assessment Centre for a previous study.

This chapter also describes the software tools which we have used during this investigation.

### ***2.2 Developmental Test of Visual-Motor Integration (VMI)***

The Beery Developmental Test of Visual-Motor Integration, or VMI, is designed to identify deficits in visual perception, fine motor skills, and hand-eye coordination. This is one of a range of tests which are administered, to assess the motor

coordination of a child referred to an assessment centre, before a diagnosis of Dyspraxia can be made.

The VMI test can be administered to a group of children simultaneously; however, it is usually administered to a single child. The child is given a booklet containing increasingly complex geometric figures. The booklet provides space for the child to copy each of the figures. Each figure is copied in turn starting with the simplest shape (see Figure 2.1) and ending with the most complex (see Figure 2.2). The test is halted when the child fails to complete three shapes in a row. During the execution of the test the booklet must not be rotated in any direction. Meulenbroek and Thomassen (1991) have shown that the accuracy of pen movements is affected by the direction of the line drawn.

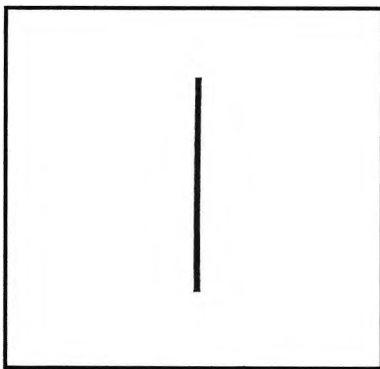


Figure 2.1 - Vertical Line

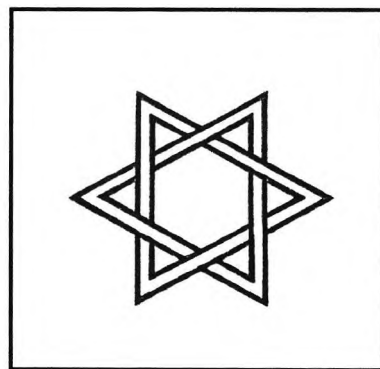


Figure 2.2 – Three-Dimensional Star

Figure 2.3 shows the complete set of 27 shapes which must be copied. In order to ensure that the child understands the task, the first three shapes can be demonstrated by the test administrator before being imitated by the child. The remainder of the shapes must all be copied without intervention.

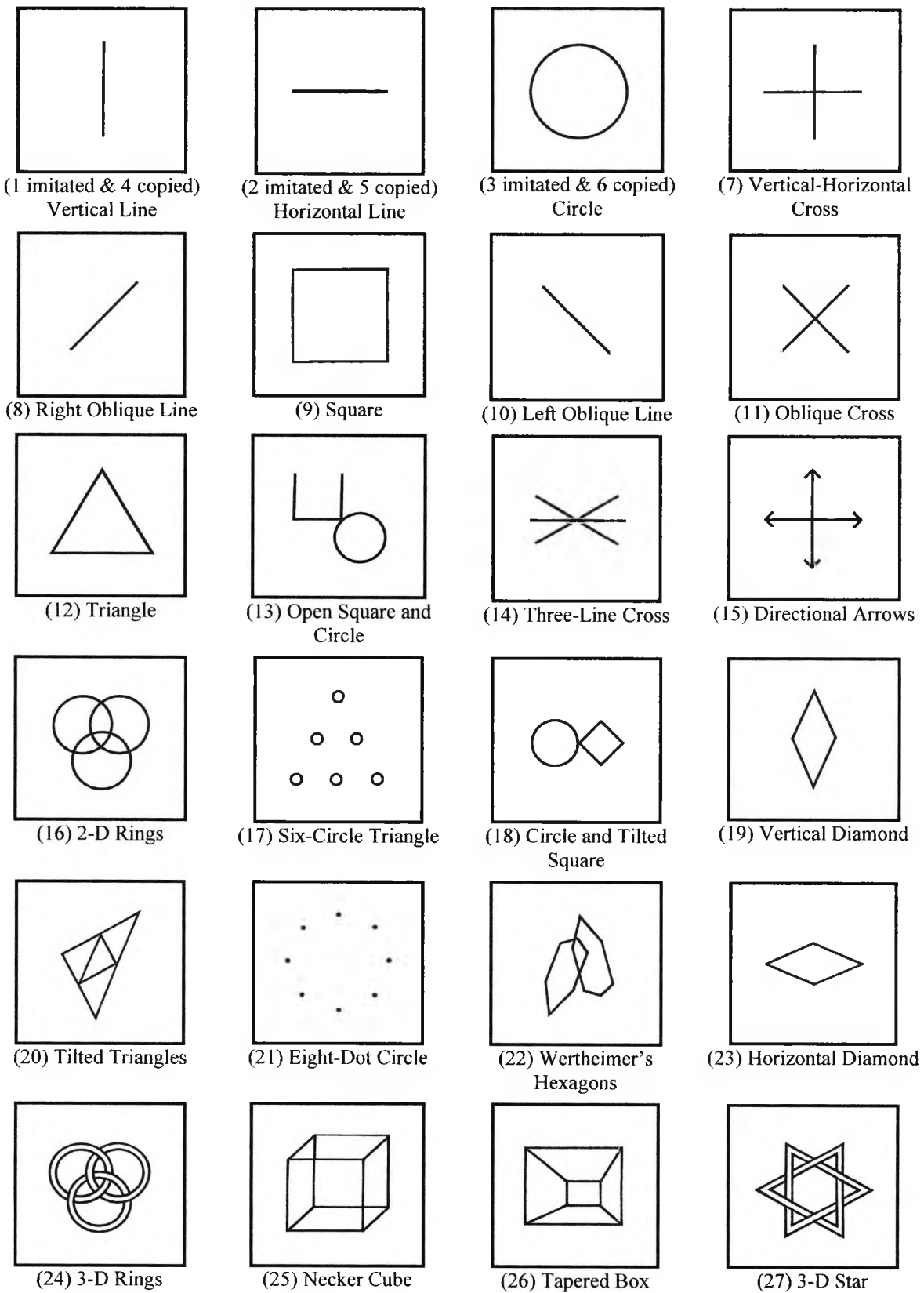
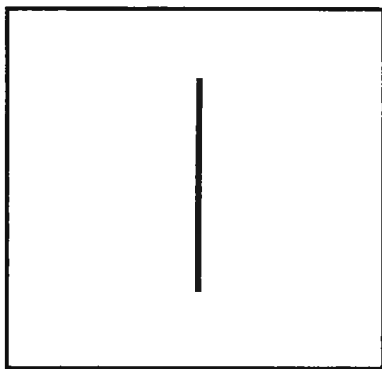


Figure 2.3 – Complete set of shapes used in VMI test

Each shape is awarded one point if it is copied successfully and zero points if it is skipped or does not meet all three of the criteria specified in the VMI manual, as noted above. If a score of zero is recorded for three shapes in a row, the scoring is halted and a score of zero is awarded to all remaining shapes, whether they are successfully copied or not. A sum of the scores for all of the shapes is then taken to produce a raw score. This is converted based on norms for each age group, and results are reported as converted scores and percentiles.

For each shape the VMI test manual defines a set of formal criteria which must be met in order for the drawing to be deemed successful. The scoring criteria for some of the shapes is shown below.



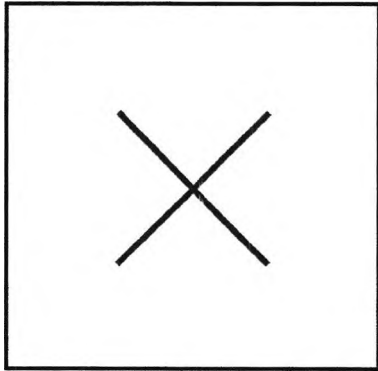
**Figure 2.4 - Vertical Line**

For the Vertical line (see Figure 2.4) there is only one criterion:

- Over half of the line(s) which make up the figure must be with 30 degrees of vertical



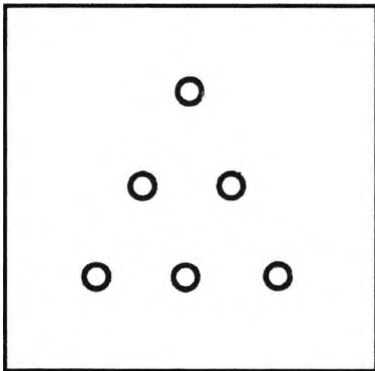
The Oblique Cross shape is shown in Figure 2.5. For this shape, three criteria are specified.



**Figure 2.5 - Oblique Cross**

Thus, the child's drawing is deemed to "pass" only if all the following three criteria are satisfied:

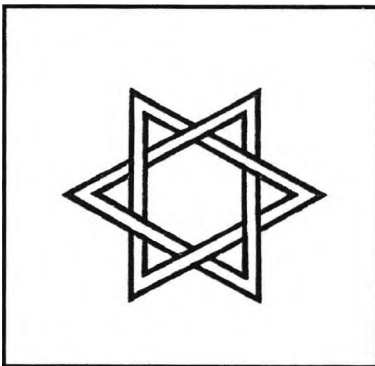
1. Two intersecting lines must be present
2. Lines angles must lie between 10-70 and 110-160 degrees.
3. Longest of the four unextended legs should be no more than twice as long as the shortest (not including extensions).



**Figure 2.6 - Six Circle Triangle**

The VMI manual specifies four criteria for The Six Circle Triangle shown in Figure 2.6. These are:

1. The shape must be made up of six circles.
2. The base and at least one other side must be straight.
3. The baseline must be within 10 degrees of horizontal.
4. The spaces between circles on the same side must be no more than 2 to 1.



**Figure 2.7 - Three-Dimensional Star**

For the final shape, the three-dimensional star (see Figure 2.7), there are three criteria:

1. All corners of triangles extend beyond opposing sides.
2. One over- and one under-lapping of the same triangle.
3. No extreme distortion.

### **2.3 Graphics Tablet**

A computerised representation of the children's drawings was created using a graphics tablet. Figure 2.8 shows the Wacom Intuos 2 tablet that was used. Wacom tablets are notable for their cordless, battery-free and pressure-sensitive pens. The absence of a battery enables the pen to closely resemble a standard writing implement. The pen we used included a functioning ballpoint tip so that drawings could be simultaneously created on paper and as data within the computer. By placing pages from the VMI test booklet on top of the tablet, we can use the standard assessment material and still capture on-line data for computer storage and processing.



**Figure 2.8 - Wacom Intuos 2 Tablet**

This tablet uses electromagnetic resonance technology so that the pen is completely passive. Under the tablet's surface is a grid of wires which send an electromagnetic signal causing oscillation in a coil-and-capacitor circuit within the pen. This signal is then switched off and the grid detects the energy of the resonant circuit's oscillations. This information is then processed to determine the position of the pen relative to the tablet. This is repeated 100 times per second to produce a time-stamped series of (x-y) coordinate points with an interval of 10ms.

By altering parameters of its oscillation circuit the pen is able to encode extra information about its current state. There are three measurements which convey further positional details about the pen, specifically:

- The angle of the pen relative to the tablet
- The direction in which the pen is tilted
- Pen tip pressure

The tablet records the position of the pen whether it is touching the tablet or not. The pen tip pressure is, therefore, especially useful as it allows us to determine when the pen is on the paper.

The data retrieved from the tablet allows us to reconstruct the drawing electronically within the computer. Geometric measurements can readily be calculated from this representation. The timing information can be used to extract dynamic information about the construction of the lines. Pen velocity and acceleration are an example of features which can easily be computed from this data, but are impossible to assess from the standard, pen-on-paper, test output.

## **2.4 Modified VMI Test**

The testing procedure for administering the VMI test must be modified in order to facilitate computer-based acquisition of the drawing data. A graphics tablet is set up on a desk which is at a suitable height for a child to sit comfortably. The graphics tablet is connected to a laptop computer which is running data capture software.

Figure 2.9 shows a typical page from the VMI test booklet. Three shapes are shown which must be copied into the three spaces below.



- Test ID: A unique number to identify the child.
- Age: Age of the child.
- Gender: Gender of the child
- Handedness: Whether the child is left or right handed.
- Status: Whether this is a known Dyspraxic child or a control subject.
- Last Medication: This is not relevant to Dyspraxia.
- HY score: This is not relevant to Dyspraxia.
- Trial: This field can be used if the same subject performs several tests

Some of these details are gathered from the child as we introduce the VMI test to them. They need to be entered into the computer before we begin the test.

## **2.5 Data Collection**

A set of data was available which had been produced from previous work within the department. This had been produced in collaboration with the Mary Sheridan Paediatric Assessment Centre (Canterbury, Kent). This data represented drawings by Dyspraxic children between the ages of 6 and 10.

In order to investigate novel methods of diagnosis, a set of drawings produced by children without Dyspraxia was required. With the help of Blean Primary School (Canterbury) the VMI test was administered to several classes of children of different age groups.

Prior to the data collection, permission slips were sent home with the children for parental approval. Children with known developmental problems were excluded from the collection. Each class group was tested over a prearranged time slot. Each child was taken individually to a separate room to perform the test; this was to prevent other children from practising the shapes before their turn. The child's details (see Section 2.4) were recorded while the test procedure was explained to them. The child was then tested using the procedure described in Section 2.4.

The focus of this data collection was to provide indications as to the feasibility of this approach. The children in the control group were not tested for Dyspraxia so it is possible that some Dyspraxic children's drawings were included. Also it is important to note that the manual scoring of the database, for both known Dyspraxic children and the control group, was performed by the author rather than a trained clinician.

In order to verify this work and produce statistically significant results it would be necessary for the test drawings to be scored professionally. Ideally this would be carried out by at least three experts so that, in case of disagreements, we would have a majority decision. It would be important to source these experts from different clinical centres to avoid bias towards the marking style of a particular centre.

Following the data collection our set of data represented drawings by:

- 90 Control Subjects
- 62 Dyspraxic Patients

The labelling of the drawings enabled the database to be processed as a whole and yet the results separated into control and patient groups.

The database has been split roughly into two halves. One half will be used as a training set, for reference and calibration whilst developing the algorithms. The other half can be used to test the software once it is completed. Each half contained the same proportion of control and Dyspraxic subjects. It is important to keep these two sets of data separate.

## **2.6 Tools Used**

During the investigation, development and documentation of this thesis, several software packages were used.

### **2.6.1 Capture Software**

The software that we used to capture data from the modified VMI test was developed within the Electronics department at the University of Kent.

### **2.6.2 Matlab**

MATLAB is a numerical computing environment created by The MathWorks. The name is an abbreviation of MATrix LABoratory. Some of its strong points are:

- Matrix manipulation
- Plotting of functions and data
- Implementation of algorithms
- Creation of user interfaces
- Interfacing with programs in other languages.

MATLAB can be used directly by typing into its command window. Also scripts can be written which can be executed with or without user intervention.

### **2.6.3 Microsoft Excel**

Microsoft Excel is a spreadsheet program which is part of the Microsoft Office suite of tools. It contains substantial mathematical functionality and allows macros to be defined to automate repetitive tasks.

### **2.6.4 Microsoft Visual Studio**

Visual Studio is an integrated development environment by Microsoft. This provides an editor, compiler and debugger for creating programs using a variety of



programming languages. Visual Studio provides a resource editing facility which allows user interfaces to be created in a simple and intuitive fashion.

The Windows API is the name given by Microsoft to the core set of application programming interfaces (APIs) available in the Microsoft Windows operating system. The functionality provided by the Windows API can be grouped into seven categories:

- **Base Services** - Provide access to the fundamental resources available to a Windows system; these include file systems, devices, processes and threads, access to the Windows registry, and error handling.
- **Graphics Device Interface** - Provide the functionality for outputting graphical content to monitors, printers and other output devices.
- **User Interface** - Provides the functionality to create and manage screen windows and most basic controls, such as buttons and scrollbars, receive mouse and keyboard input, and other functionality associated with the Graphical User Interface (GUI).
- **Common Dialog Box Library** - Provides applications with standard dialog boxes. For example, opening and saving files or choosing colour and font.
- **Common Control Library** - Gives applications access to some advanced controls provided by the operating system. These include status bars, progress bars, toolbars and tabs.
- **Windows Shell** - Allows applications to access the functionality provided by the operating system shell, as well as change and enhance it.
- **Network Services** - Gives access to the various networking capabilities of the operating system. Its sub-components include NetBIOS, Winsock, NetDDE, RPC and many others.

The Microsoft Foundation Class Library (MFC) is a software library produced by Microsoft which is also included in Visual Studio. The MFC wraps portions of the Windows API in C++ classes, forming an application framework. Classes are defined for many of the handle-managed Windows objects and also for predefined windows and common controls. The MFC allows a more object oriented way of interacting with the API. This simplifies creation of Windows programs by providing automatic closure of handles when the objects creating them go out of scope. It also provides utility classes and collection classes

### **2.6.5 Microsoft Word**

Microsoft Word is a word processing package which is part of the Microsoft Office suite of tools.

### **2.6.6 Paint Shop Pro**

Paint Shop Pro is a graphics package from Corel. It is a bitmap graphics editor and vector graphics editor. It facilitates the creation and manipulation of images and it also features a screen capture facility.

### **2.6.7 UMLet**

UMLet is an open source UML editor. It is Java based and has been designed for quickly creating UML diagrams.

## **2.7 Software Design**

All of the software that has written to support this thesis has been written using Microsoft Visual Studio. All of the code was written in C++. The programs have been designed using the MFC and compiled for a 32-bit Windows environment.

All functions have been designed to be easily readable. Comments to describe each procedure and consistent formatting have been used throughout.

## **2.8 Summary**

In this chapter we have introduced the Beery Developmental Test of Visual-Motor Integration (VMI), which is the movement assessment test that the work in this thesis is primarily based upon. We have described how this test has been adapted to produce not only a paper based result, but also a computer representation of the drawings. This modified test has been used to augment existing data with drawings produced by children at a local school.

We have also listed the software which was used during this work.

The next chapter describes a system we have developed which allows a clinician to perform scoring of the VMI test on a computer rather than using a pen and paper.

# Chapter 3

## Assisted Scorer

### ***3.1 Introduction***

We have shown in the previous chapter how drawings produced from the modified VMI test are captured and stored on a computer. This chapter shows how a clinician's job of scoring the VMI test can be improved using this data.

Specifically, a computerised system has been devised which could be used to assist the traditional manual scoring of a completed VMI test by a clinician (typically, an occupational therapist or similar). The conventional assessment of a child through the VMI test requires that each copied figure in the test is examined and assigned a "score" according to a very strictly defined set of prescribed rules. The score can be used in various ways, but is generally used to derive an "age-equivalent" for the child being tested. Our aim was to exploit the availability of on-line captured representations of the child's output to provide novel computer-based support to improve the convenience, efficiency and accuracy of the scoring process, to the benefit of both the clinical value of the results and the effect of the testing on individual patients.

The approach proposed offers several advantages over the simple pen only based test. The accuracy of the scoring is improved by allowing the clinician access to visual

tools which enable accurate measurement of angles and distances on the drawings. This in turn reduces the variability in scores returned by different clinicians. The children's drawings can be transferred electronically which is not only much quicker than posting but is also a lot more practical. A test booklet from the VMI test is over 20 pages long so transporting a set of tests for an entire school year would be costly and also difficult to manage. With this system the drawings could be sent via email or more securely on a CD or memory card.

A database format has been designed to store, in a single file, the drawing information of a set of children. One of the design criteria for this was to preserve as much of the captured data as possible, even if it is not needed for scoring the VMI. For example, movements performed with the pen off the paper and pen orientation are inherently preserved within the capture process. This is so that in future processing, or in research, the data for these children can be re-used without having to spend time building another database.

There are two steps to scoring the VMI test output using the software developed as an implementation of the proposed technique.

1. Loading the drawings into the software
2. Scoring the drawings

These steps can be carried out at the same time or separately. Loading the drawings could be performed during the data collection; the single data file could then be emailed to a clinician for scoring. The data file does not have to be created in one go, and can be extended at any time. The task of importing the data could therefore be split into several smaller sessions. Databases can be combined so this could even be done by a team of people. This would be especially useful in a school situation where data from different classes may be collected by different people.

The software would first be used to create a database of the children's drawings. This could be carried out by the person supervising the test. Figure 3.1 shows the initial screen of the software. As subjects are added to the database they are shown in a list on this screen. A small summary of the scoring progress and some details for each

patient are also displayed. From this screen it is also possible to remove patients from the database and load an existing database from a file. Once the database has been created it can be saved to disk ready for scoring.

The screenshot shows a window titled "VMI Scoring Assistant" with a menu bar (File, Test, Help) and a toolbar. Below the toolbar is a tabbed interface with "Summary" selected and "Forms" tabs numbered 1 through 27. The main area displays a table with the following columns: Index, ID, Status, Gender, Age, Handednes, Loaded, Scored, Passed, and VMI Score. The table contains 32 rows of patient data.

Index	ID	Status	Gender	Age	Handednes	Loaded	Scored	Passed	VMI Score
1	1	Control	Male	7	Right	24	24	18	21
2	3	Control	Female	7	Right	24	24	15	18
3	5	Control	Male	7	Right	22	22	17	20
4	7	Control	Male	7	Right	24	24	19	18
5	9	Control	Female	6	Right	23	23	17	19
6	11	Control	Male	7	Right	24	24	12	13
7	13	Control	Female	7	Right	24	24	16	19
8	15	Control	Female	7	Right	24	24	10	11
9	17	Control	Male	7	Right	24	24	16	19
10	1	Control	Female	8	Right	24	24	19	22
11	3	Control	Female	9	Right	24	24	19	22
12	5	Control	Female	9	Right	21	21	16	3
13	7	Control	Male	8	Right	24	24	16	16
14	9	Control	Male	9	Right	24	24	16	19
15	11	Control	Male	8	Right	21	21	15	18
16	13	Control	Female	8	Right	24	24	20	23
17	15	Control	Female	6	Right	18	18	11	14
18	17	Control	Female	5	Right	18	18	14	17
19	19	Control	Female	6	Right	21	21	12	15
20	21	Control	Male	6	Right	18	18	13	16
21	23	Control	Male	6	Right	21	21	14	17
22	25	Control	Female	9	Right	24	24	20	23
23	27	Control	Female	10	Right	24	24	19	22
24	29	Control	Female	10	Right	24	24	23	26
25	31	Control	Female	9	Right	24	24	16	19
26	33	Control	Female	9	Right	24	24	17	20
27	35	Control	Female	9	Right	24	24	13	15
28	37	Control	Female	9	Right	24	24	14	16
29	39	Control	Male	8	Right	24	24	13	16
30	41	Control	Male	8	Right	24	24	17	18
31	43	Control	Male	8	Right	24	24	13	16
32	45	Control	Male	5	Right	15	15	10	13

Figure 3.1 List of imported patients

The modified VMI test procedure captures three drawings at a time (see Chapter 2). Before they can be scored the individual drawings must be separated. Figure 3.2 shows the graphical interface which assists splitting these into the individual drawings. While importing data we can choose to remove any extra lines which may have been accidentally drawn during the VMI test procedure. For example, the child may have touched the pen against the paper either before or after the test had finished.

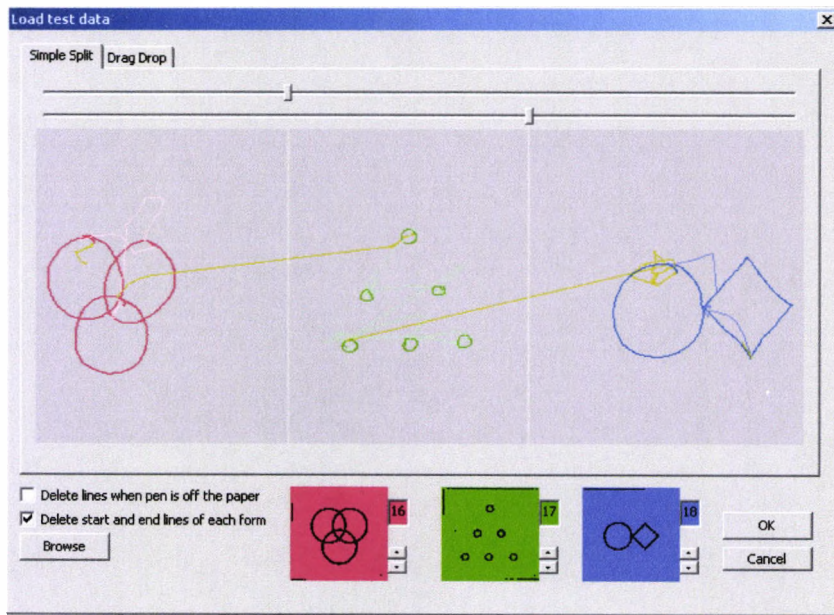


Figure 3.2 Importing data from modified VMI

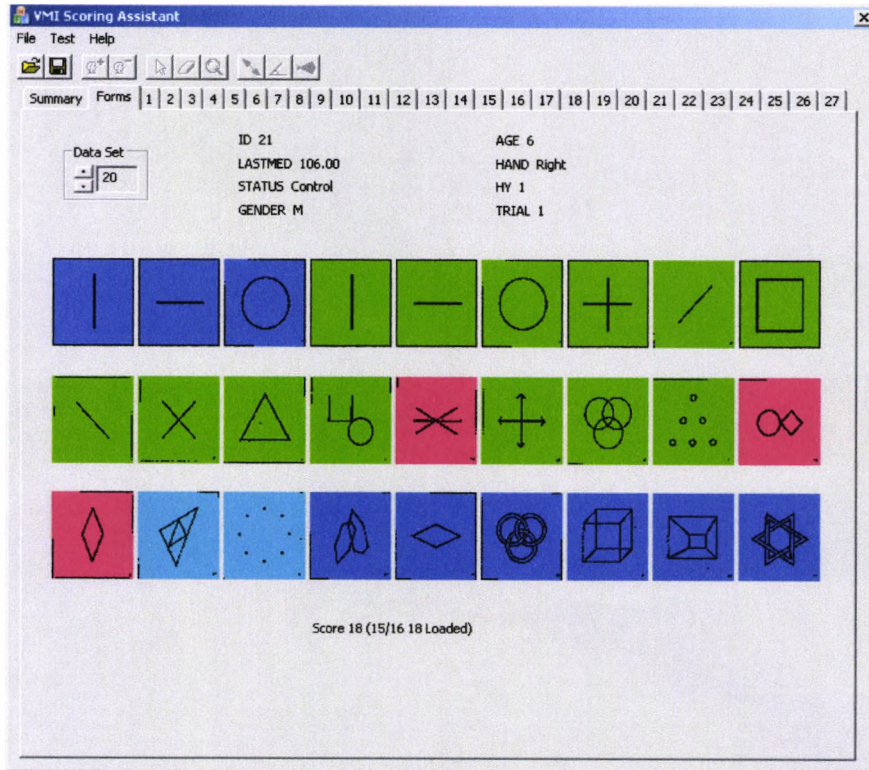


Figure 3.3 Patient Summary

When the clinician comes to score the database they will employ the same software as was used for creating the database. They will however use much more of the functionality provided by the software. Figure 3.3 shows the summary page for one patient; here the clinician can quickly see the scoring progress of all of the patient's drawings. Figure 3.4 shows the view of a single drawing; from this page the clinician can perform measurements on the drawings. A zoom function is available which allows the user to get a closer look for more accurate measurements.

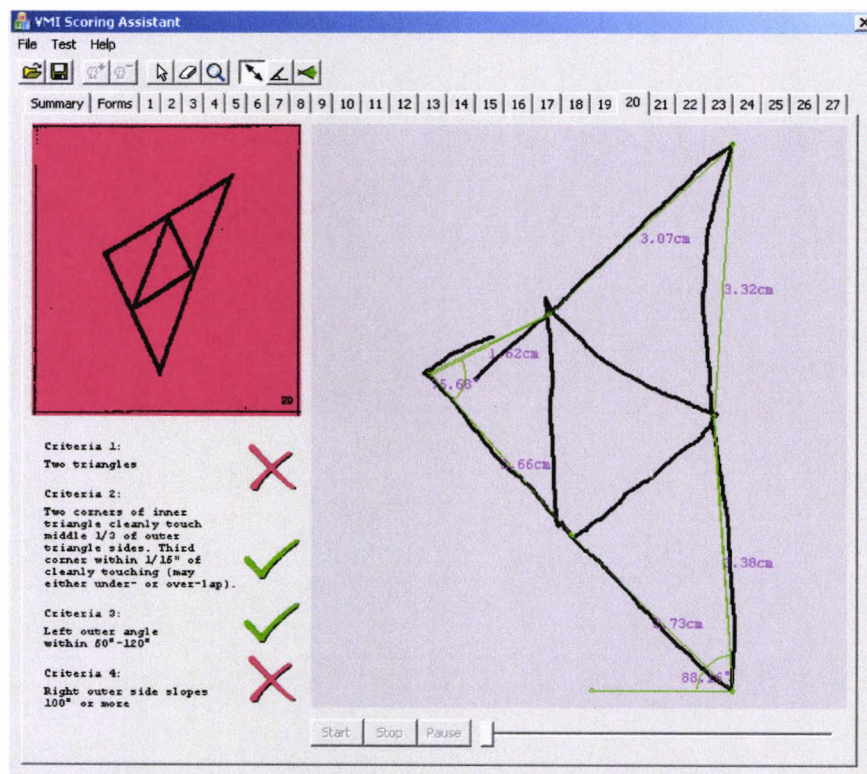


Figure 3.4 Single Shape with measurements

In order to give the clinician a better understanding of how the drawing was produced, it is possible to play back the movements of the pen in real time. The animation produced reveals the construction method and can often give an insight into some of the less intelligible drawings.

For each drawing the ideal shape is shown along with a list of the criteria by which it should be scored. The shape can be scored by simply clicking on the appropriate criteria to toggle between pass, fail and not scored. The overall score for the shape will be automatically worked out from these criteria. This is then displayed by colour coding the background of the ideal shape. See Figure 3.4.



After the drawings have been scored the database will keep a record of which scoring criteria have passed and failed for each shape. This can be saved to a separate file for future reference and may be used to compare progress to a later test.

A summary of the scores can be exported in a format readable by most spreadsheet programs.

### **3.2 State Manager**

The functionality of the software is largely driven by the user interface. Keeping track of all of the elements of the user interface became increasingly complex as more features were added to the software. A class was designed to manage the state of the system. In each state different screen elements are made visible or invisible and are enabled or disabled depending on what we want the user to see and be able to do. For example, until the user selects a patient for study, the software prevents them from trying to display which shapes have been attempted.

Figure 3.5 shows the 11 possible states in which the system can exist, and the possible state transitions among them. Dashed lines show transitions which can only happen in one direction.



Figure 3.5 VMI Assist State Diagram

The 11 states can be grouped into 3 sets of similar displays, as illustrated in Table 1.

Group	Description
Database View	<p>The screen displays a list of all the patients which are loaded into the system. Some details of each patient are shown along with the scoring progress.</p> <p>There are four states that fall within this group, specifically:</p> <ul style="list-style-type: none"> <li>• Database Empty</li> <li>• Nothing Selected</li> <li>• Patient Selected</li> <li>• File Access</li> </ul> <p>These states are described in Section 3.3.</p>
Patient Summary	<p>In this state the scoring progress of a single patient can easily be assessed. Idealised pictures of the full set of VMI shapes are displayed, colour coded depending on the scoring progress.</p> <p>There is only one state associated with this group, this is described in Section 3.4.</p>
Single Shape	<p>The screen displays full details of the drawing of a single shape. The patient's drawing is reconstructed here along with the ideal shape from the VMI booklet and a list of the scoring criteria.</p> <p>There are six states that fall within this group, specifically:</p> <ul style="list-style-type: none"> <li>• No Data</li> <li>• Loaded</li> <li>• Zoom Selected</li> <li>• Tool Selected</li> <li>• Animation Running</li> <li>• Animation Paused</li> </ul> <p>These states are described in Section 3.5.</p>

**Table 1 - State groupings**

### 3.3 Database View Page

Figure 3.6 shows the database view page. This is the first page the user will see when the program is started.

The screenshot shows a window titled "VMI Scoring Assistant" with a menu bar (File, Test, Help) and a toolbar. Below the toolbar is a tabbed interface with "Summary" selected and tabs for forms 1 through 27. The main area displays a table with the following data:

Index	ID	Status	Gender	Age	Handednes	Loaded	Scored	Passed	VMI Score ▲
1	1	Control	Male	7	Right	24	24	18	21
2	3	Control	Female	7	Right	24	24	15	18
3	5	Control	Male	7	Right	22	22	17	20
4	7	Control	Male	7	Right	24	24	19	18
5	9	Control	Female	6	Right	23	23	17	19
6	11	Control	Male	7	Right	24	24	12	13
7	13	Control	Female	7	Right	24	24	16	19
8	15	Control	Female	7	Right	24	24	10	11
9	17	Control	Male	7	Right	24	24	16	19
10	1	Control	Female	8	Right	24	24	19	22
11	3	Control	Female	9	Right	24	24	19	22
12	5	Control	Female	9	Right	21	21	16	3
13	7	Control	Male	8	Right	24	24	16	16
14	9	Control	Male	9	Right	24	24	16	19
15	11	Control	Male	8	Right	21	21	15	18
16	13	Control	Female	8	Right	24	24	20	23
17	15	Control	Female	6	Right	18	18	11	14
18	17	Control	Female	5	Right	18	18	14	17
19	19	Control	Female	6	Right	21	21	12	15
20	21	Control	Male	6	Right	18	18	13	16
21	23	Control	Male	6	Right	21	21	14	17
22	25	Control	Female	9	Right	24	24	20	23
23	27	Control	Female	10	Right	24	24	19	22
24	29	Control	Female	10	Right	24	24	23	26
25	31	Control	Female	9	Right	24	24	16	19
26	33	Control	Female	9	Right	24	24	17	20
27	35	Control	Female	9	Right	24	24	13	15
28	37	Control	Female	9	Right	24	24	14	16
29	39	Control	Male	8	Right	24	24	13	16
30	41	Control	Male	8	Right	24	24	17	18
31	43	Control	Male	8	Right	24	24	13	16
32	45	Control	Male	5	Right	15	15	10	13

Figure 3.6 Database View Page

Whilst displaying the Patient Summary Page the state manager can be in four possible states. Which state it is in determines which operations can be performed.

**Database Empty** – The user can only load a database or add a new patient.

**Nothing Selected** – The user can load or save the database and add a new patient. A patient can be selected by clicking on the appropriate line in the list.

**Patient Selected** – The user can perform the above operations and also use the tab box to go to the Shape Summary Page or any of the Single Shape Display pages for the selected patient. The currently selected patient can also be removed from the database.

**File Access** – This is a special state that the software goes into whenever the database file is being accessed or the current database is cleared. In this state all the screen elements are disabled, the user cannot click on any buttons. While the database is being accessed a progress bar and cancel button are presented in a separate window. Once the file access has completed the system always stays on the patient summary page, either in None Loaded or None Selected state.

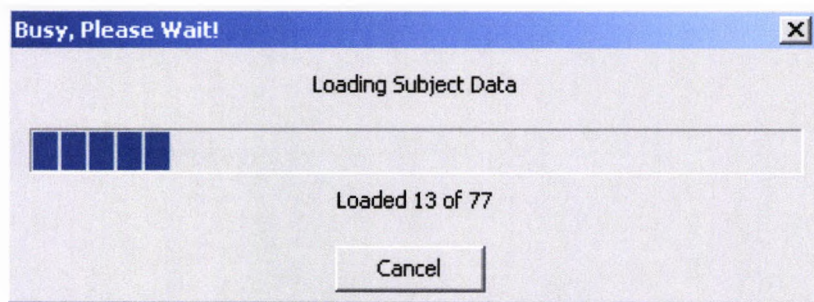
### 3.3.1 Database Management

Loading and saving the database to disk is a time consuming process. In order to prevent the program becoming unresponsive, we created a multi-threaded mechanism to allow the user to interact with the system whilst the database operations are being performed.

A new thread is created which performs the database action. Meanwhile the original thread simply ensures that the program window continues to be updated and listens for user input.

The separate tasks communicate via windows messaging allowing the interface process to halt the progress of the database task. Also the database task sends messages to the interface task after each subject is loaded; this causes it to update the progress bar.

Figure 3.7 shows the pop-up box which is displayed whilst the database information is being read from a file. This provides the user with a progress bar and also allows the user the option of cancelling the task. The text message on the pop-up is changed for other database actions. It is also displayed for saving to disk and freeing resources when the database is closed.



**Figure 3.7 Loading the database**

While the database file is being accessed the state manager moves to the 'Database File Access' state. The main window shows the list of loaded subjects. All the interface options in the main window are disabled. This prevents the user from altering data which may no longer be valid. Until the database action is completed the only thing the user can do is cancel the operation. If cancel is clicked the program will roll back the action to a state where it can reliably continue.

### **3.3.2 Debug Screen**

Whilst developing the software it was found that a debug output would be very useful for tracking down bugs and observing the internal workings of the system. This evolved into an extra section of the display which could be enabled or disabled from the help menu. This is not affected by the state manager, except for being disabled during database access, and remains visible in all display states.

Figure 3.8 shows the main window with the debug screen enabled. For a release version of the software this option would be disabled and the menu item removed.

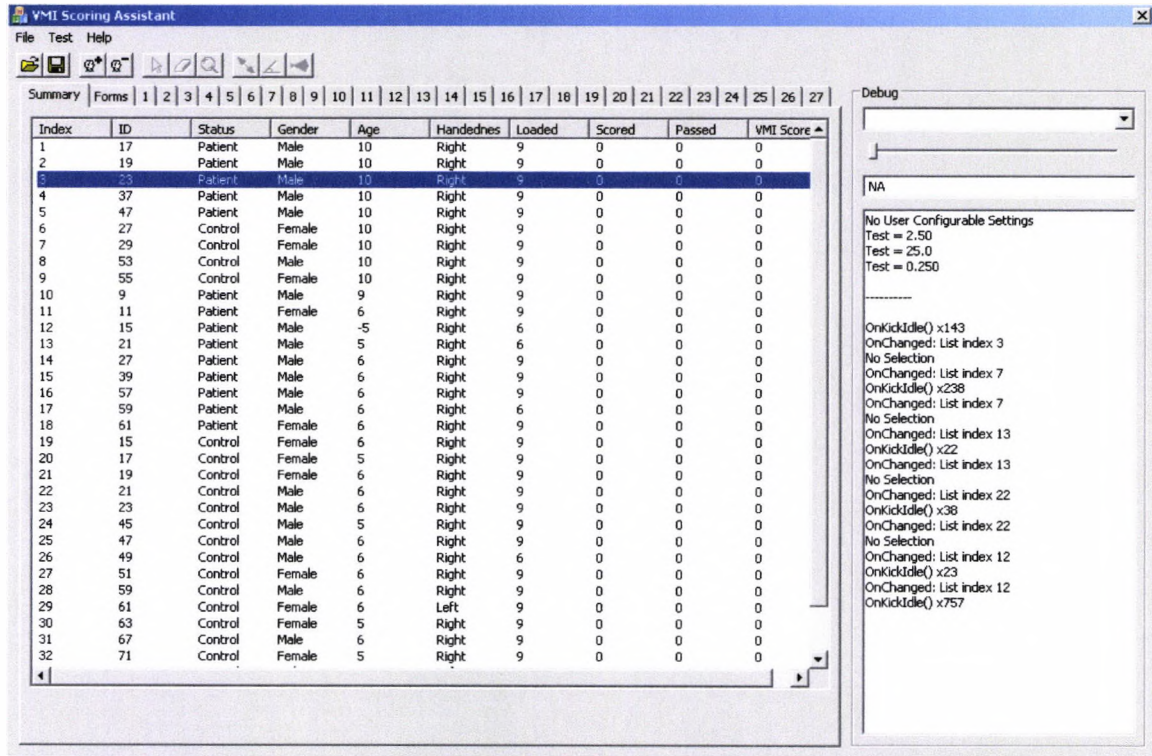


Figure 3.8 Debug Screen

The system can output text to the debug screen at any time. Also the debug screen has been designed to allow the user to change settings from here.

### 3.4 Patient Summary Page

Figure 3.9 shows the patient summary page. This page gives an overview of the scoring status of a single patient along with information about the patient. The shapes from the VMI manual are displayed; these are colour coded to allow for quick visual inspection. Specifically, we adopt the following convention:

- Green – indicates that the shape has “passed” according to the designated criteria.
- Red – indicates that the shape has “failed” one or more of the designated criteria.
- Dark blue – indicates that patient did not attempt to draw the shape.
- Light blue – indicates that a drawing exists but has not yet been assessed.

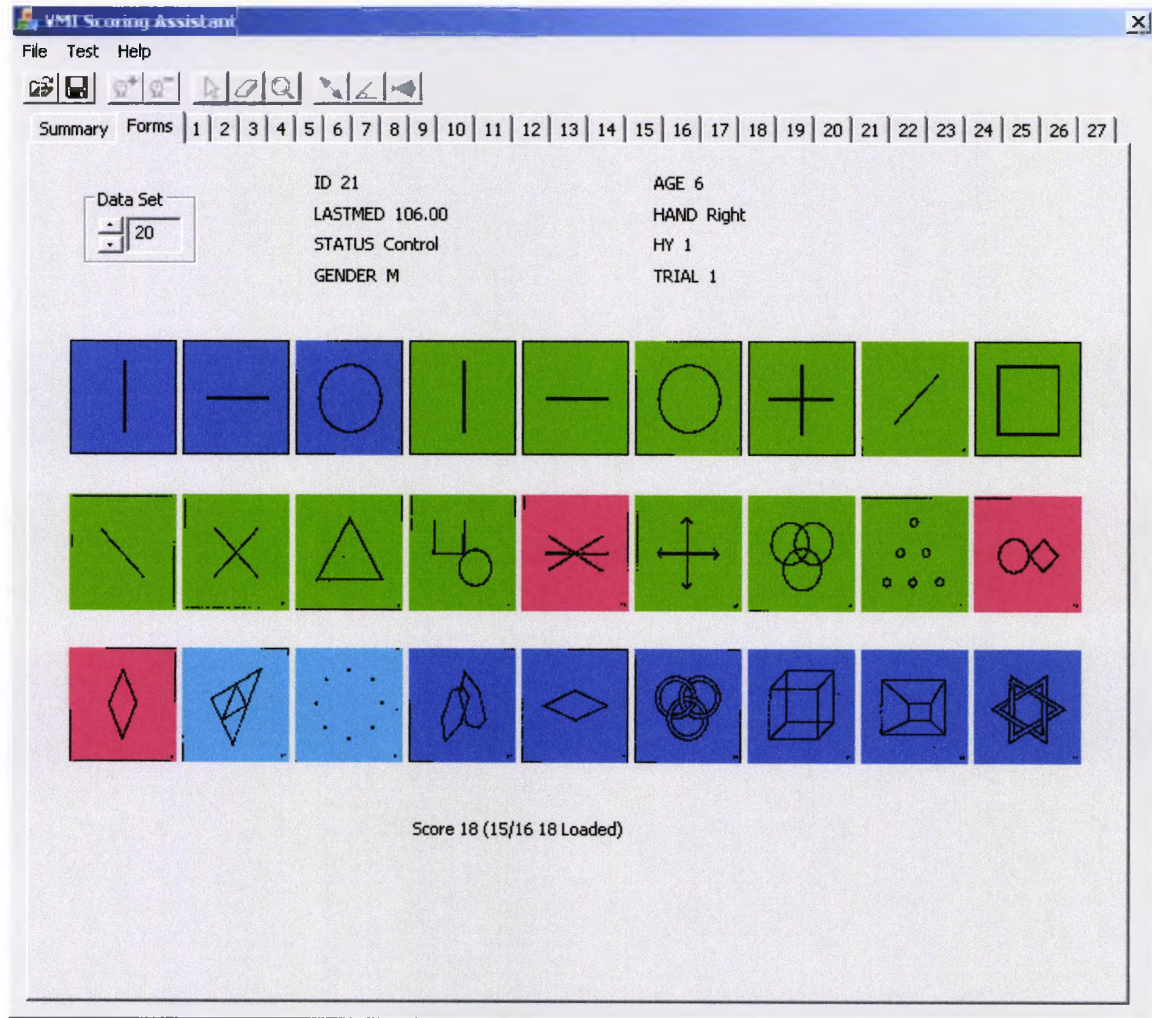


Figure 3.9 Shape Summary Page

The user is brought here from the database view page and can return here from looking at a single shape.

The Patient Summary Page has one state. The user can click on the tabs to change to a different page. The numbered tabs take them to the respective shape in the VMI test. Clicking on the pictures also takes them to the page for that shape.



### 3.5 Single Shape Display

Figure 3.10 shows the Single Form Display page. This page shows all the information about the drawing of a single shape. The largest area of the screen shows the patient's drawing; both on paper and off paper movements are displayed. Next to the patients drawing is shown the ideal shape, and underneath it the list of scoring criteria is displayed.

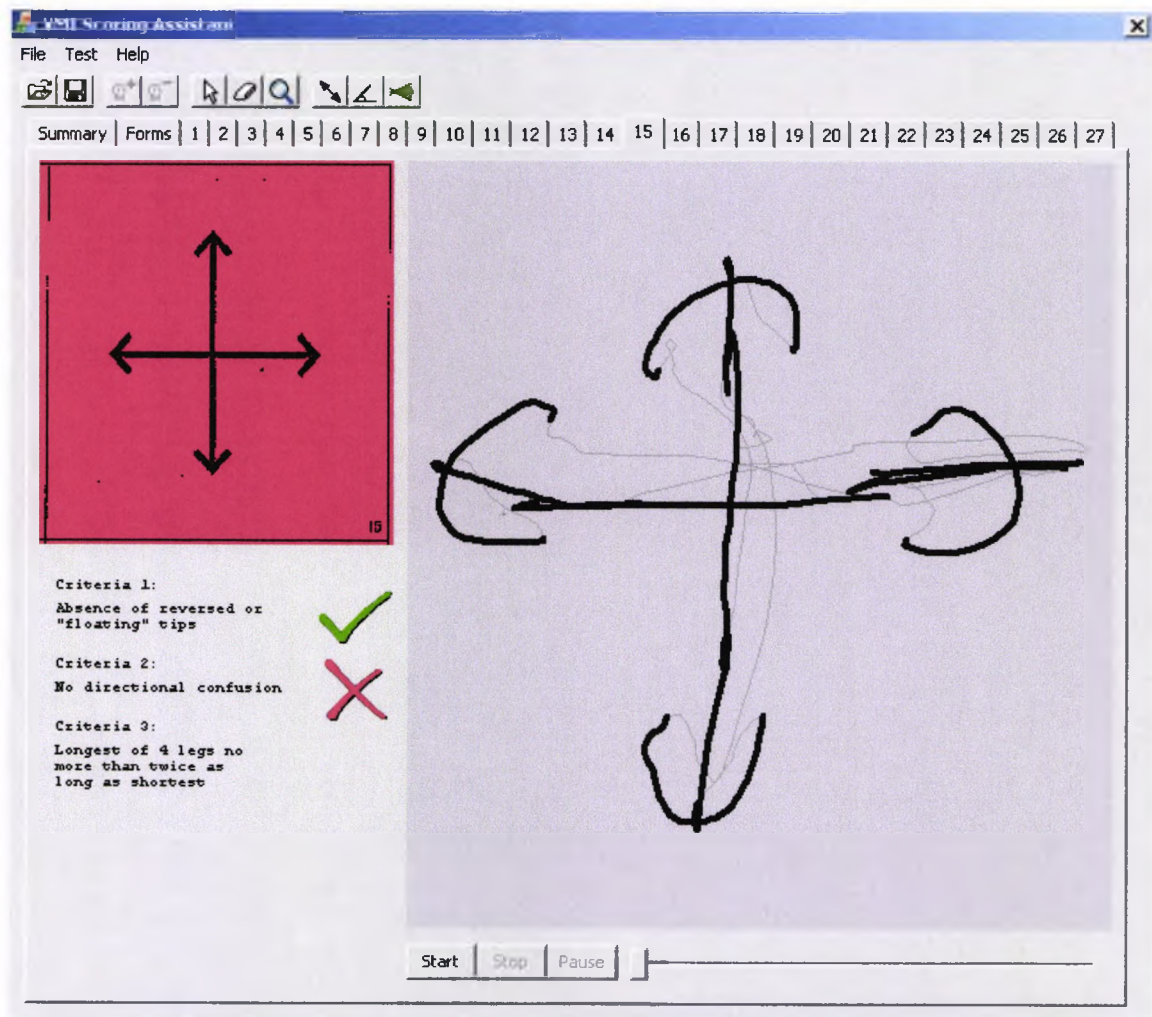


Figure 3.10 Single Form Display

Clicking on the scoring criteria causes toggling between “pass”, “fail” and “not scored”. A tick, cross or nothing is displayed respectively. Failure against any one criterion is enough for the whole shape to fail. In order for the whole shape to pass all criteria must be satisfied. The background colour of the ideal shape changes when

enough information has been provided to get an overall score; green for a pass or red for a fail.

This screen is the most complicated and the state manager can be in one of 6 different states.

**No Data:** All of the features are disabled except for the tab buttons which can be used to move to a different screen. From the file menu, the user can load drawing data into the database.

**Loaded:** This is the state shown in Figure 3.10. All the features are available. The user can select tools to use from the tool bar. Also the start button on the animation controls can be pressed to play an animation of the drawing sequence

**Animation Running:** This state is entered when the user clicks on the start button. The animation of the child's drawing is running. The 'Stop' and 'Pause' buttons are enabled and the slider bar moves across the screen to show the position through the animation.

**Animation Paused:** The animation is paused. This state can be entered by clicking pause when in the 'Animation Running' state, or from the 'Loaded' state by dragging the slider bar. The slider bar can be dragged to skip through the animation and find a particular frame. Whilst in this state the text on the pause button changes to 'resume'; clicking this starts the animation running from the current position.

**Tool Selected** – When a tool is selected the animation controls are disabled and the user can click on the drawing to take new measurements.

**Zoom Selected** – This state allows the user to zoom into the shape for a closer look.

### 3.5.1 Zoom

The system allows the user to zoom into the drawing for closer inspection of more complex features. Whilst zoomed the tools are still visible and new tools can be used. The animation can also be run whilst zoomed.

Figure 3.11 shows a drawing which has been zoomed to check the scoring criteria. In this shape the scoring rule being invoked is “corner within 1/16 of an inch of cleanly touching (may either under- or over-lap)”. The zoom display makes this easy to measure.

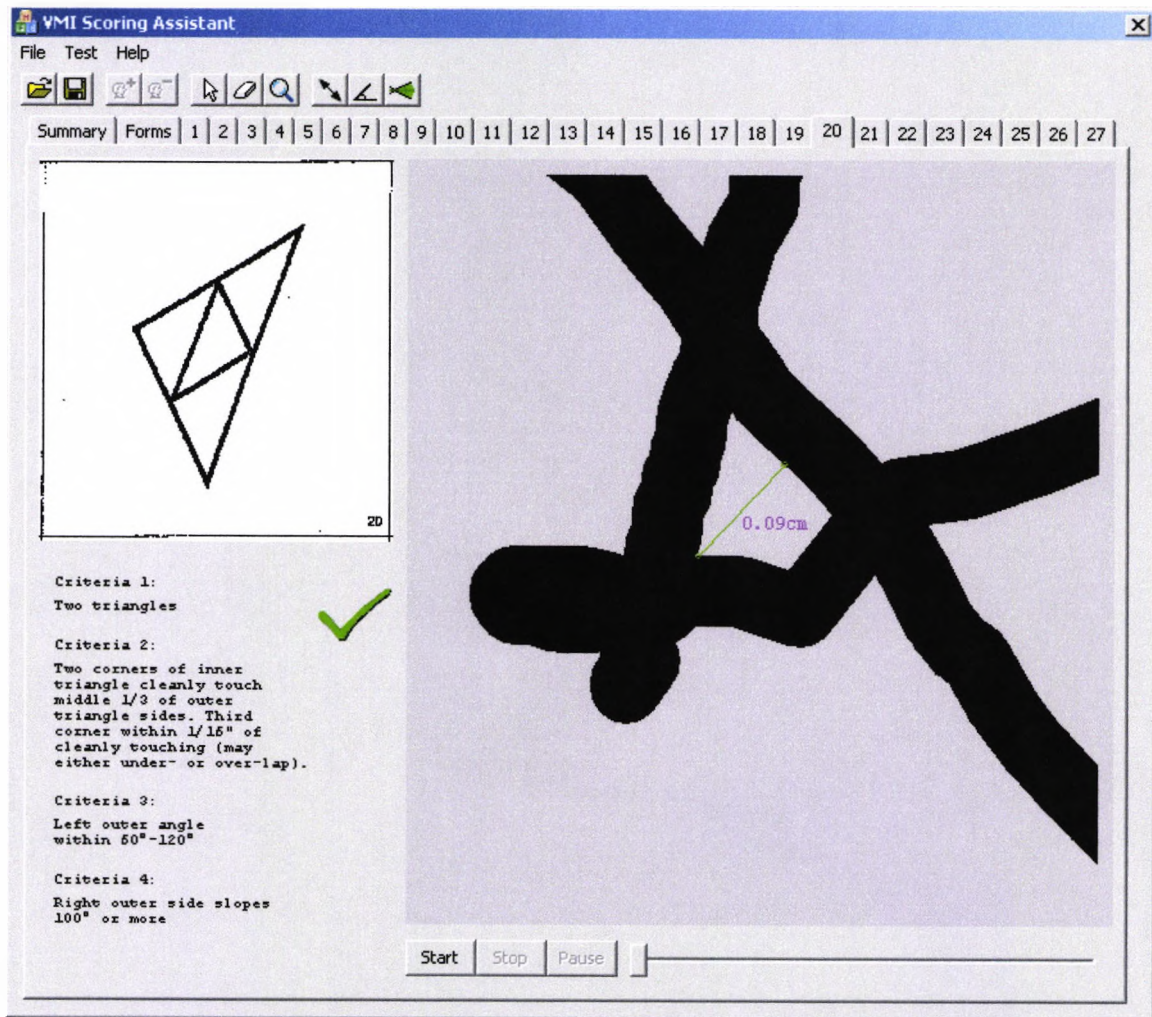


Figure 3.11 Zoom into drawing

When the zoom button is clicked the cursor will change to represent a magnifying glass whilst over the image display. By clicking the left mouse button the image will be zoomed in and by clicking the right mouse button the image will be zoomed out. The keyboard cursor keys can be used to scroll the image whilst zoomed in. The image can be zoomed into repeatedly for an even closer look. The maximum zoom level is only limited by the memory capacity of the computer. Effective “magnification” of up to 10x is easily achievable. In order to display the zoomed image the whole drawing is rendered at a higher resolution, a sub image from this is then displayed.

### 3.5.2 Image Display

In order to have a flicker free image display we have used image buffering. Rather than rendering the image directly onto the screen, it first renders it into a memory buffer and then copies this to the screen.

The various components of the final image are stored in separate buffers.

**zoomBitmap:** an image buffer which contains the full patient’s drawing scaled to the current zoom level. All of the tools which have been finished are also drawn onto this. If a tool is still in use this is drawn later.

**dataBitmap:** the section of the zoomBitmap which currently is being displayed. When the image is not zoomed this buffer contains exactly the same data as zoomBitmap.

**toolBitmap:** if a tool is currently in use, this buffer is used to draw it. The dataBitmap image is copied to this, and then the partly finished tool is drawn on top of it. The tool is continuously redrawn as the mouse is dragged around and this extra buffer vastly improves the refresh rate.

When the image on the screen is refreshed the dataBitmap or toolBitmap is already prepared with the new image, so we simply copy them to the screen.

The zoomBitmap is created so that the user can scroll around the zoomed image. When the cursor keys are pressed we simple copy a different section of the zoomBitmap into the dataBitmap. Without this buffer we would have to re-render the dataBitmap from scratch which would result in a very slow and jerky scroll.

### 3.5.3 Tools

One of the main advantages of the computer assisted scoring process is the ability to accurately measure aspects of the drawings. This is achieved through the use of tools.

Using the various tools we can draw guides onto the display. The user can use several tools at once. The user can go back and edit the tools after they have been finished. Also the tools can be deleted from the display.

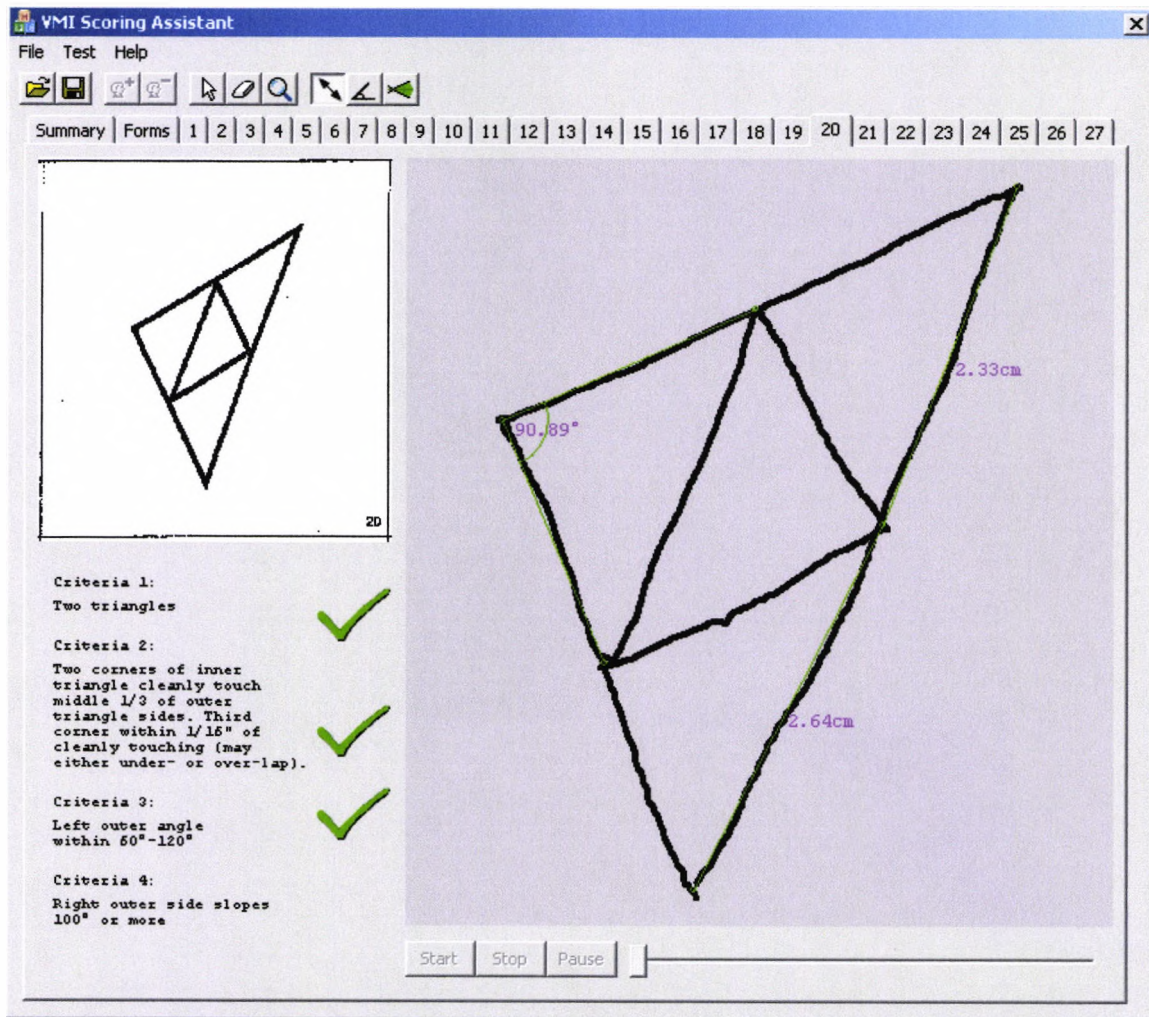


Figure 3.12 measurement tools

Figure 3.12 shows how the measurement tools can be used to accurately determine whether the drawing passes the criteria. The angle tool has been used to test Criteria 3 “Left outer angle within  $60^{\circ} - 120^{\circ}$ ”. Two length tools have been used to work out whether the inner triangle hits within the middle third of the outer triangle, to test one part of Criteria 2 “corners of inner triangle touch middle 1/3 of outer triangle sides”.

To activate a tool the user simply clicks on the appropriate icon on the tool bar. The user then clicks on the drawing to add the points which make up the tool. The length tool requires two points, one at each end. The angle tool requires three points.

Once tools have been added they can be adjusted in place. After selecting the pointer tool the user can click on the image display. The nearest tool point will be selected and can be dragged to a new location. This can be performed even after zooming into the image where the entire tool may not be visible.

The tool system was designed to be easily updateable. If a clinician suggests a new tool which would make scoring the VMI easier, it can be quickly and easily added to the toolkit. Each tool is defined in a class which is derived from a common base class. The base class provides all of the functionality for positioning the control points.

### **3.5.4 Animation**

For animation the rendering process is altered slightly. Instead of joining together all of the samples to create a line, the function just draws one section of the line at a time. This function is called every 10ms to reconstruct the drawing in real time.

When the animation is started, the zoomBitmap (see Section 3.5.2) is cleared and all the tools are rendered to it. As the animation runs, each section of the line is added incrementally to this image buffer. The visible section of this buffer is then copied to the dataBitmap (see Section 3.5.2) ready to be in turn copied to the screen. It is not possible to add new tools while the animation is running so the toolBitmap (see Section 3.5.2) is not used.

The timing is done using a Windows timer event handler. If the computer is running another program in the background it may not have time to update the screen before the event is triggered again. The update mechanism has been designed to cope with this. If the computer is too busy to update the display Windows will not perform the refresh. In this case the timer event updates the dataBitmap (Section 3.5.2) only.

When the computer passes control back to the software it will process all of the time requests and display the latest frame.

During the animation the child's drawing is rendered over the top of the tools. This is not perceived to be a problem. To rectify this would require either re-rendering the tools for each animation frame, or, using an extra buffer for the tools which would be merged on top of the drawing. Merging buffers is much slower than simply copying data between buffers. Either of these operations would be computationally expensive.

### **3.6 *Importing Data***

The modified VMI test (see Section 2.4) is performed using a separate data capture software package. The output from this program is a set of several data files for each subject. Each data file contains the sample points for up to three of the shapes. Each file also contains a header section which holds patient information. This information is entered manually by the clinician at the time of the test.

Before we can import the drawings into the program database we first need to split the data into individual shapes. Also we need to determine, for each shape, the corresponding shape in the VMI test.

A standard Windows open file dialog is used to find and select the data file. When the user opens the file a new dialog box appears. This provides the user with a graphical interface which allows the data to be easily split into three separate shapes.

In the majority of cases, separating the three shapes can be done simply by dividing the data up with a straight line. Sometimes however the lines from one shape can overlap over the lines from another shape and a more sophisticated method is needed.

### 3.6.1 Simple Split Dialog

Figure 3.13 shows the dialog box which pops up to divide drawing data into three separate shapes. The user can drag the scroll bars near the top of the dialog box to determine where the division between the different shapes will occur. The lines are shown in red, green or blue to indicate whether they have been allocated to the first, second or third shape respectively. As the scroll bars are moved the colour coding of the lines is updated in real time to reflect the new position of the split points.

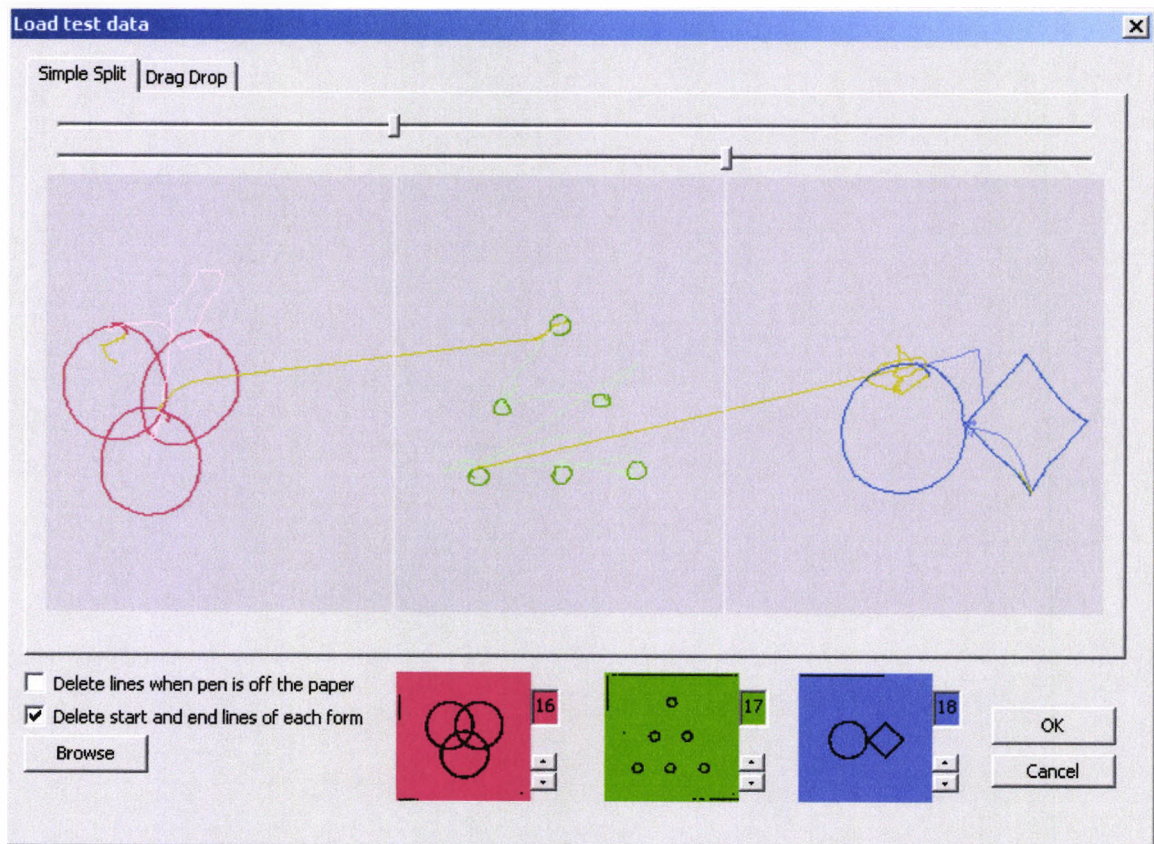


Figure 3.13 Dialog box for dividing subjects drawing

The user can select the VMI shapes that the patient's drawings represent. This is useful when the file contents are unknown. Colour coded pictures of the ideal shapes are shown to give a visual aid to the scorer. The colours match the colour of the lines which will be assigned to that shape. The three shapes that are initially displayed depend on which shape the user is viewing when the dialog is opened.



Lines where the pen was off the paper are shown in a lighter shade of the appropriate colour. The user can select whether these pen up lines are stored in the database. The lines between shapes are removed by default, but this can be changed if the user wishes to preserve them. Lines which will not be added to the database are shown in brown.

By clicking the Browse button the user is presented with the standard windows load dialog box which can be used to change the data file which is being loaded.

### 3.6.2 Drag Drop Split

For some drawings the pen movements of two or more of the forms can overlap. In this case a simple split position is not enough to separate the form data. By clicking on the 'drag-drop' tab at the top of the dialog box the display changes as shown in Figure 3.14. The lines that make up each of the shapes are show in separate boxes. A fourth box is also displayed which shows lines which will not be added to the database.

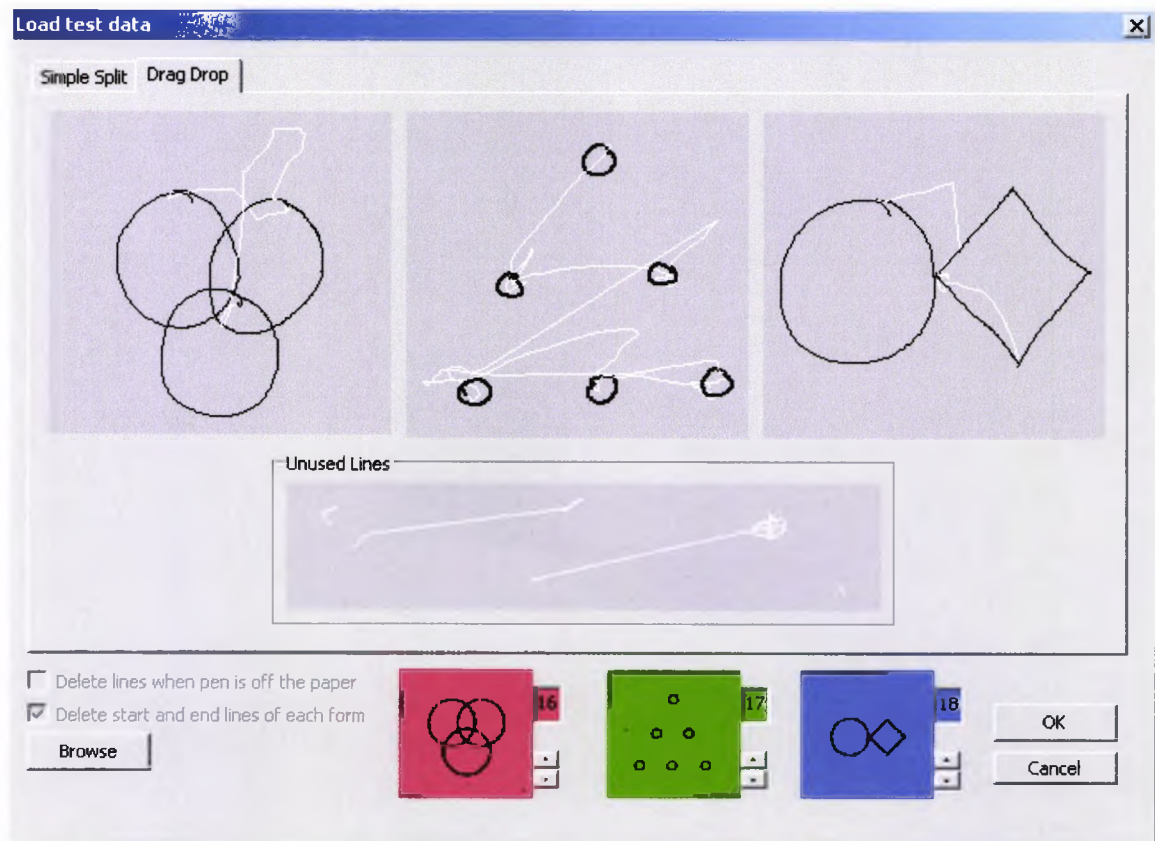


Figure 3.14 Drag Drop

The user can now click on the lines making up a shape and drag by holding down the mouse button. When the mouse button is released over one of the other boxes, the line joins that set of lines. Lines dragged to the Unused Lines box will not be added into the database.

This can also be useful for cleaning up a drawing. If pen movements have accidentally been recorded on the tablet before the start of the drawing, these extra lines can be easily removed.

Upon returning to the 'simple split' tab, the new segmentations will persist unless the slider bars are moved, or the check boxes are changed. Pen-down lines which have been marked as unused, will be colour coded in black.

### 3.6.3 Data Format

The data associated with each patient can vary in size depending on how many shapes have been drawn and, within those shapes, how many sample points have been used to capture them. In order to accommodate such a widely varying set of data a class hierarchy has been used.

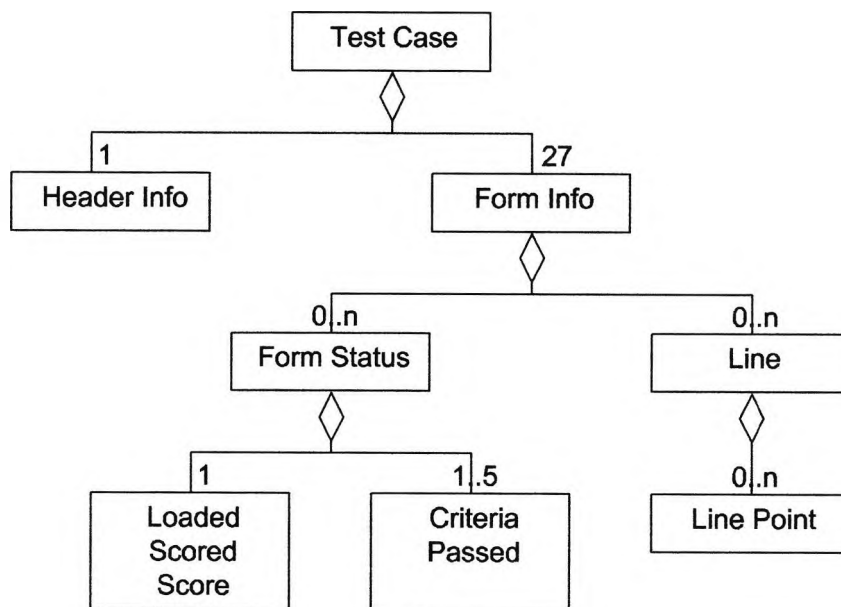


Figure 3.15 Class Diagram of subject data storage

In the main program we simply have a vector of `TestCase` objects. The rest of the structure is then hidden and accessed through this vector. Figure 3.15 shows the elements of the data structure that is used to store the data.

As well as storing the data, this class structure also contains a lot of functions to assist with organising, maintaining and displaying the data.

`FormInfo` is a base class which is specialised for each of the different shapes. Each sub class sets up the appropriate text for the VMI criteria display and allocates the correct number of 'Criteria Passed' objects.

The line object contains all the data points that represent a single line drawn by the subject. A pen-down line is taken from when the pen touches the paper until it leaves the paper again. Pen-up lines can also be stored.

The `LineElement` object contains all the data for a single data point captured by the graphics tablet, specifically:

- Timestamp
- X Position
- Y Position
- Pressure
- Altitude
- Azimuth

The pressure is used to determine whether the pen is on or off the paper. The altitude and azimuth are not used by the software but are stored for possible future reference. By retaining all the data even if it is unused the database produced by the software will be much more useful in future work.

## 3.7 File Output

### 3.7.1 Database Format

The database is saved to disk in a proprietary format (see Figure 3.16). The format was designed to be human readable and as easy to understand as possible. It is hoped that once a database has been created using this program it will be used by future projects. This would avoid having to segment the forms again. For this reason we store all the information which has been captured from the graphics tablet even though some of it will not be used.

The parameters used for the data capture are also stored in the file format. Along with the patient details we store the sampling frequency and horizontal and vertical resolution. Table 2 describes how these parameters are represented in the format.

Fs	This is the data rate, specified as the number of samples per second. For our data this was 100.
XRES	The horizontal resolution of the recorded data. For our data this was 1000 per cm, which is one hundredth of a millimetre. This is the equivalent of 2540 dpi.
YRES	The vertical resolution. As for XRES this was one hundredth of a millimetre for our data.

**Table 2 - Data capture parameters**

The timestamp, *time*, associated with each data sample is specified in milliseconds. Therefore, for the data collected during this research, the samples will be separated in intervals of 10. The horizontal, *x*, and vertical, *y*, position of each sample is measured from the top-left of the tablet with a resolution according to XRES and YRES respectively.

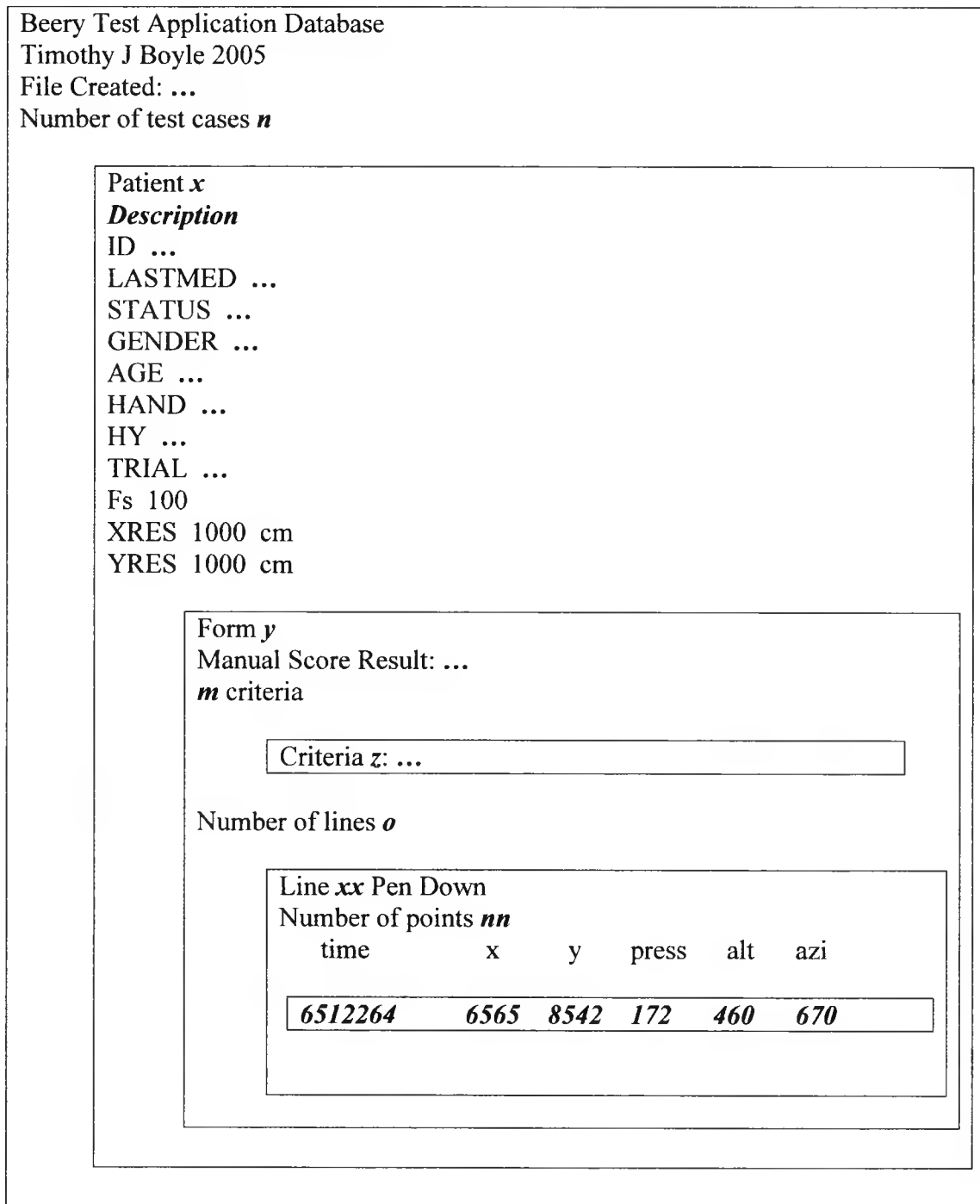


Figure 3.16 Database File Format

### 3.7.2 Results Output

As well as being able to save the results into the database, the results of the scoring process can also be saved as a separate text file. The elements are ‘tab-separated’ so that the file can be easily loaded into Microsoft Excel or another spreadsheet program. Once in Excel it is a simple matter to colour code the results and apply a table format.

Shape	1	2	3	4	5	6	7		21	22	23	24	25	26	27
ID 17	NL	NL	NL	PS	PS	PS	PS		PS	FL	FL	FL	NL	NL	NL
ID 19	NL	NL	NL	PS	PS	FL	FL		PS	PS	PS	PS	FL	FL	FL
ID 23	NL	NL	NL	PS	PS	FL	PS		PS	PS	PS	FL	PS	FL	FL
ID 37	PS	PS	PS	PS	PS	PS	PS		FL	NL	NL	NL	NL	NL	NL
ID 47	PS	PS	PS	PS	PS	PS	PS		PS	PS	PS	FL	FL	FL	FL
ID 75	NL	NL	NL	PS	PS	PS	PS		PS	FL	PS	FL	FL	FL	FL
ID 77	NL	NL	NL	PS	PS	PS	PS		PS	PS	FL	PS	PS	FL	FL

Figure 3.17 - Colour coded summary of shape scoring

Figure 3.17 show a cut down version of the colour coded output for a set of patients.

The cells of the chart are encoded as:

“PS” (Green) – The shape has been scored as “pass”.

“FL” (Red) – The shape has been scored as “fail”.

“NL” (Blue) – The shape has not been scored.

### **3.8 Conclusion**

This chapter has described a computer system which was developed to assist a clinician to perform scoring of the Beery Developmental Test of Visual-Motor Integration (VMI).

A graphical user interface has been provided to allow the clinician to quickly and easily assess a set of VMI drawings. Improvements to the objectivity of scoring have been achieved by allowing the clinician access to visual measurement tools. The administration of remote scoring has been made more convenient by encoding the set of completed test booklets into a single data file.

This software has been implemented using Visual C++.

The next chapter moves the discussion forward by investigating software which could produce a VMI test score without requiring a trained clinician.

# Chapter 4

## General Scorer

### ***4.1 Introduction***

While developing an automated implementation of the VMI test it became apparent that a lot of the functionality and scoring procedure would be shared across more than one shape. A flexible system was needed that could be used to analyse the whole set of shapes rather than a separate approach for each one.

A novel generic solution was thus investigated, resulting in a system which lends itself to analysis of any copied figure. This system uses three separate stages:

1. Identifying the points of interest (POI).
2. Matching the points found to a template.
3. Applying a set of rules.



### 4.1.1 Identifying the points of interest

Finding the points of interest in the drawing is mainly done using simple geometry. Corners are identified by finding points on the lines where the pen velocity drops below a threshold. Typical examples of some of the possible points of interest include:

- Corners
- Intersections (lines cross)
- Touching Lines
- Line Ends

Around a feature such as an intersection there may be several related points. To either side of an intersection there may, for example, be points where the lines touch. Clustering is applied to reduce these related points down to a single point of interest. Some of the clustered points may be centred on a drawing artefact rather than a feature of the shape, these clusters are removed.

The lines that connect the clusters are also recorded so they can be matched to a template.

### 4.1.2 Matching the points found to a template

A template is produced based on the ideal form in the VMI manual. The template is a list of rules defining how points relate to each other. Each rule specifies two points and the relationship between them. The template can specify connections and also direction. The domain of possible solutions can be searched to find which points in the sample match each point in the template.

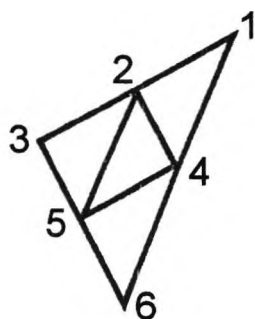


Figure 4.1 - VMI Shape 20

Figure 4.1 shows a shape from the VMI test. The features of the shape have been numbered. These are the points of interest that we want to detect with the first step of the process. Examples of template rules for this shape might be:

- Point 1 connects Point 2.
- Point 3 is above Point 6.

Enough rules need to be specified to ensure that only one match is likely to be found for a set of detected points.

### **4.1.3 Applying a set of rules**

For each shape a set of criteria is specified in the VMI manual. These can be broken down into a smaller set of general tests to apply to a drawing. We refer to these general tests as ‘rules’.

The clusters we have detected mark out important features of the child’s drawing. We can imagine that a simplified version of drawing could be created by joining the clusters together with straight lines. Some rules can be tested by measuring angles or distances on this skeletal version of the drawing. These measurements can be performed knowing only the positions of the clusters. The simplified drawing therefore does not have to be physically created.

Some of the rules cannot be verified from the clusters alone. Tests for these need to be carried out on the underlying drawing data.

Each rule returns a simple pass or fail response. In the majority of cases all of the rules must pass in order for the shape to meet all of the required VMI criteria. It is also possible to label a subset of the rules and specify how many of them need to pass. For example, in the shape 20 (see Figure 4.1), the VMI requires that two of the internal corners must touch the outer triangle. We can set a rule for each of the corners and specify that only two of these three rules needs to pass.

### 4.1.4 Complete System

Initially, software was implemented to perform each of these three stages separately. This was quite cumbersome to use as it required files to be saved and loaded between each stage. The three programs were therefore combined into a single software component which could perform all of the functions required.

The scoring software was implemented as a component which can be included into any Visual Studio project. This enables the scoring functionality to be easily added to other programs. It is hoped that eventually the VMI scoring assistant (see Chapter 3) will be able to incorporate automatic shape scoring.

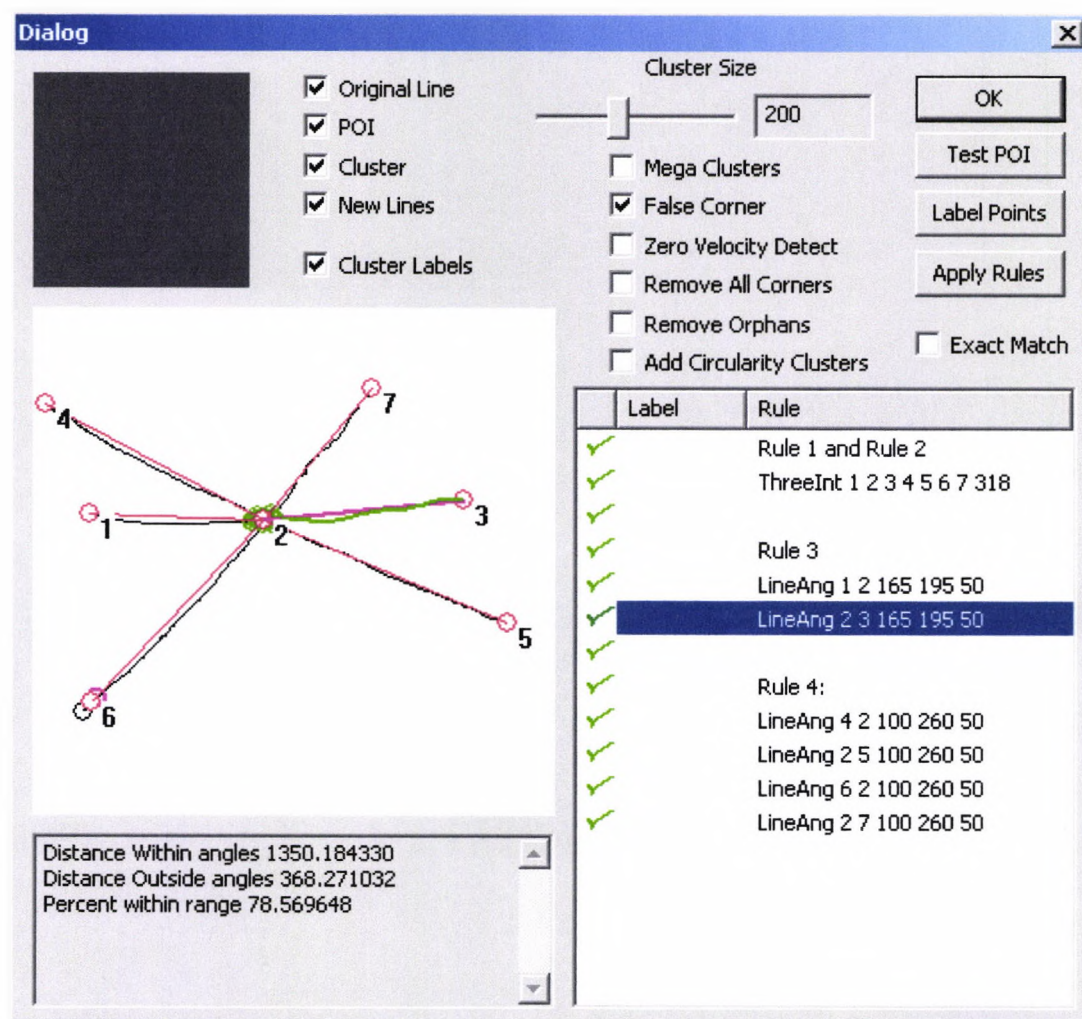


Figure 4.2 - Scoring system interface

Figure 4.2 shows the graphical user interface for the automatic scoring system. The component can be run with or without displaying the dialog window. When the dialog window is displayed various options are available to the user.

As the scoring software has been implemented as a component, it needs to be included in an executable program in order to be run. A test harness was created which allows the user to assess the VMI criteria for a series of drawings. The user interface (see Figure 4.3) allows configuration of all components external to the scorer itself.

Drawing data can be loaded in from a database created with the VMI assist software (see chapter 3), or directly from the data collected from the modified VMI test (see chapter 2). Template data can also be loaded from a database or built up from the files used by the original three programs.

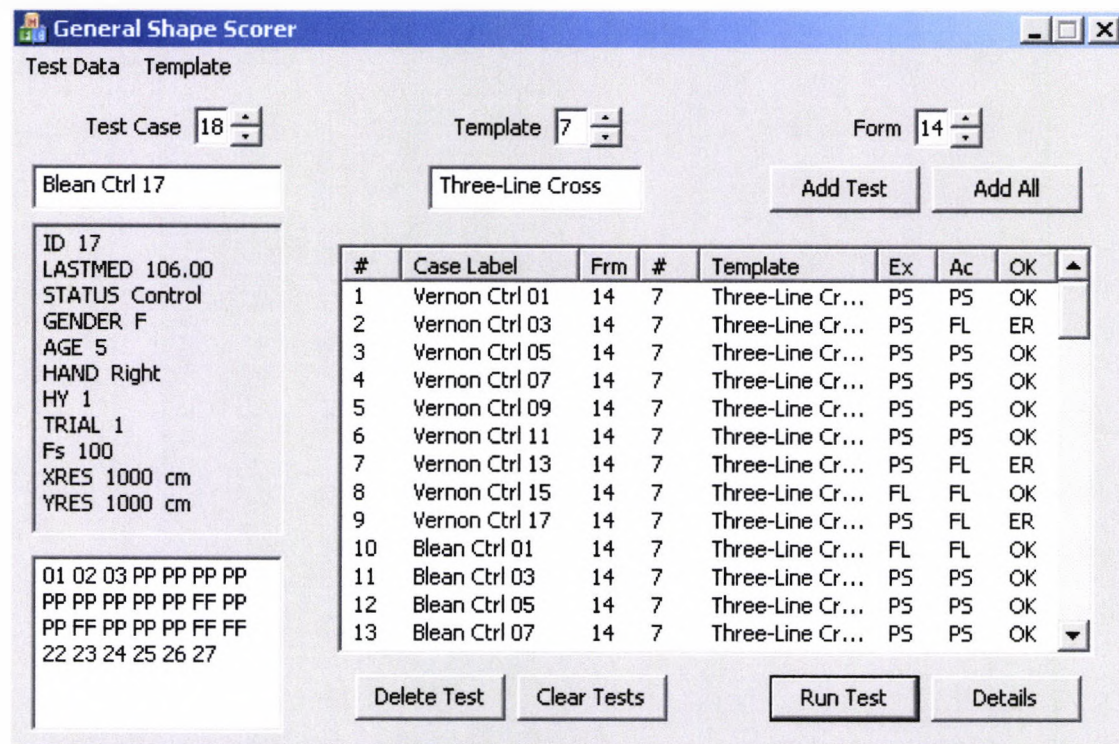


Figure 4.3 - Test Harness

The test harness allows the user to select a shape number and a template to test against. This combination is then added to a list of tests to perform. The user can add the drawing from just the currently selected patient or the same shape for all loaded patients.

The database of subjects can store a verified score for each shape. This scoring must be performed by a qualified clinician. For testing purposes the scoring has been approximated.

The box in the bottom left of the window (see Figure 4.3) shows the shapes completed by the currently selected patient. If data is available it is marked PP, FF or XX depending on whether it is correct, incorrect or not scored respectively. If no data is available the shape number is displayed instead.

The main list box in the window (see Figure 4.3) shows a list of all the scheduled tests. If the test has been performed the result is compared against the verified score. The last three columns of the window show:

- Ex: The expected result, ideally produced by a clinician.
- Ac: The actual result produced by the software (if the test has been performed)
- OK: Indicates if the expected result and actual results match.

The comparison between expected and actual results can be used to indicate the scoring accuracy of the software. Figure 4.4 shows a dialog window which was implemented to show an overview of the currently loaded test set.

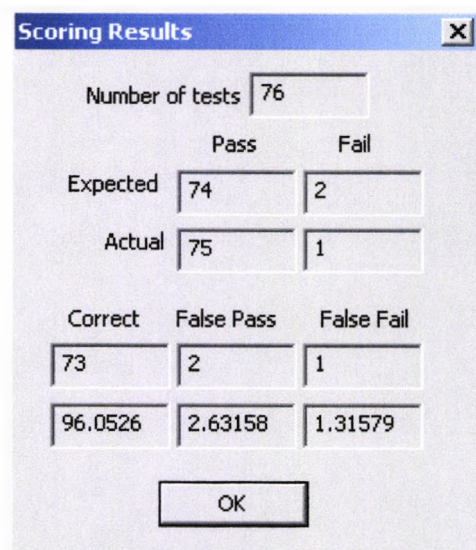


Figure 4.4 - Scoring Results

## 4.2 POI Identification

The purpose of the POI Identification step is to describe the child's drawing as a set of coordinates and a list of lines connecting them. The coordinates represent important features of the shape. If we plotted the coordinates and drew in the connecting lines we would produce a greatly simplified version of the child's drawing.

There are 4 main steps in the process.

1. Finding the initial features
2. Clustering
3. Finding connections
4. Removing unnecessary points

These steps are described in more detail in the following sections.

### 4.2.1 Implementation

The class diagram for the POI Identification stage is shown in Figure 4.5.

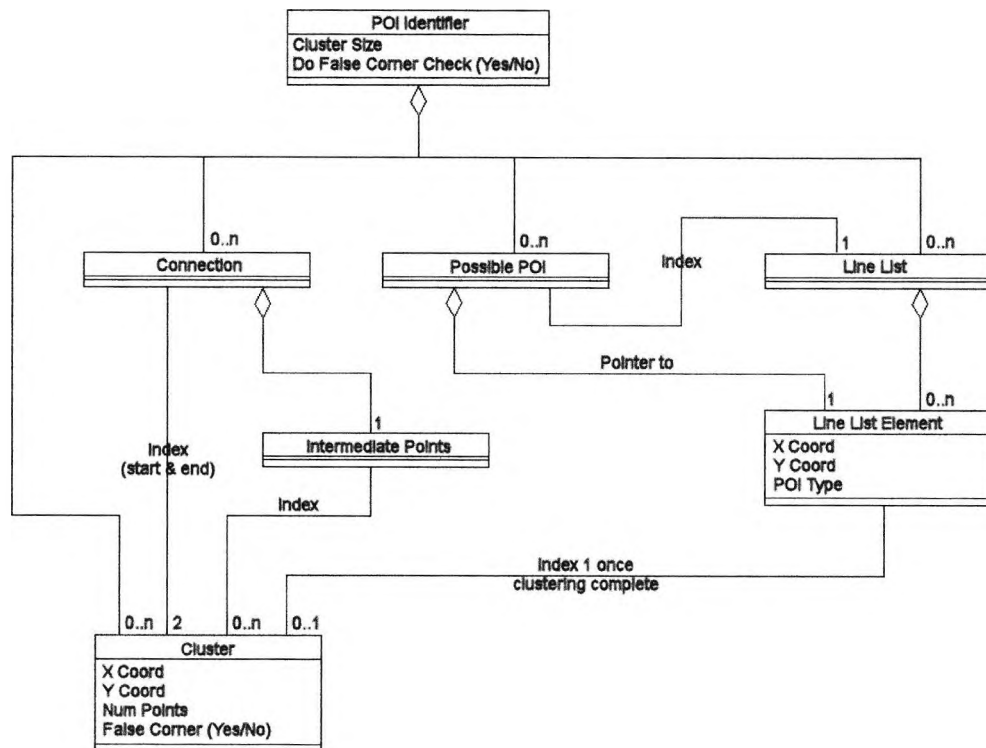


Figure 4.5 - POI Identification Class diagram

The `LineList` and `LineListElement` classes have been imported from the Assisted Scoring software (see Chapter 3). Some extra data members have been added to these classes to store the points of interest found. These are:

`POIType`: The type of interest point. This is an integer value which can hold the following values:

0. Not a POI
1. Line Ends
2. Intersections (lines cross)
3. Touching Lines
4. Corners
5. Removed Point

`clusterNumber`: The cluster the point belongs to:

This is also an integer. In the class diagram this is the index between the `LineListElement` class and the `Cluster` class. The index is zero based so the `POIType` must be checked to determine whether the `LineListElement` is actually a POI.

To show how the data is processed, we will highlight parts of the class diagram as sections are populated with data. During the loading of the shape into the program, the `LineList` and `LineListElement` classes are filled. Figure 4.6 shows the class diagram in the initial state. Note that the POI Type attribute of the `LineListElement` is not set at this stage.

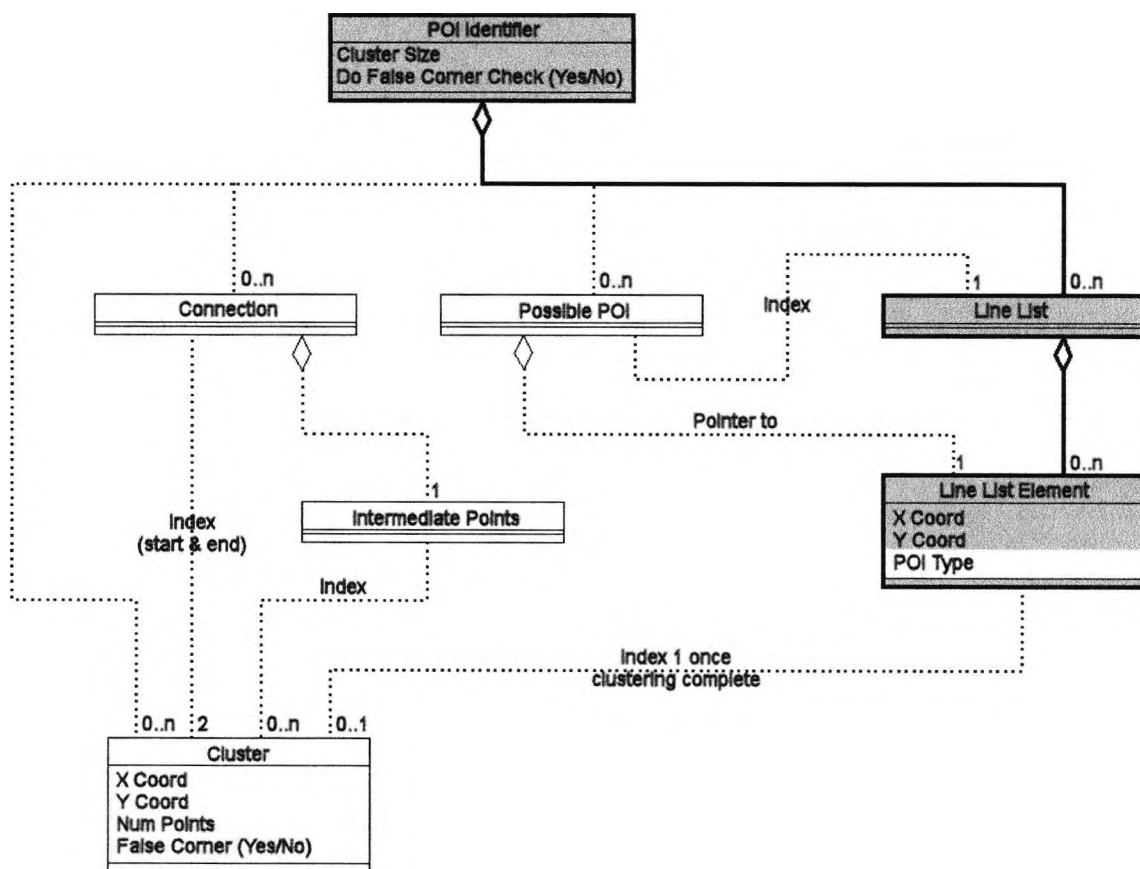


Figure 4.6 - Initial State

### 4.2.2 Initial interest points

Finding the initial interest points is done in two steps.

First the lines are scanned to find the interest points. When interest points are found the line element is flagged with the type of point it represents. This is described in more detail in Section 4.2.2.1.

Secondly the list of interest points is populated. This is done by scanning through the line elements a second time and creating an interest point whenever a feature is flagged.

The reason for doing this in two steps is that an element may contain more than one feature. For example, at an intersection the element will not only be marked as an ‘intersect’ but also that it is touching another line. If we added a point each time we



found a feature we would have to search through the list of points to prevent duplication. If more than one feature is found for an element, the element type is determined by the feature which is most important. An 'intersect' is a very strong feature which can be accurately placed. 'Touching lines' is a feature which tends to be spread across many neighbouring elements. Therefore, in this example, the element would be flagged as an 'intersect'.

After completing this stage the POI Type attribute of LineListElement in Figure 4.6 will have been set.

### **4.2.2.1 Scanning lines**

The line features we are interested in are:

1. Line End
2. Intersect (lines cross)
3. Touching
4. Corner

#### **4.2.2.1.1 Line Ends**

As we scan through the line the ends are simply the first and last point in the list. These are easily found.

#### **4.2.2.1.2 Intersects and Touching**

'Intersects' and 'Touching lines' are found by stepping through every section of every line and comparing it with all sections from all the other lines.

##### **4.2.2.1.2.1 Formula**

Intersections are found by simultaneous equations of the form  $y=mx+c$  for the two line sections and the checking that the results fall within both lines.

Touching lines are found by drawing an imaginary circle around each point in the line and checking the intersections of that circle with each section of the other lines.

There are some special cases that need to be dealt with, specifically:

- Vertical lines (standard formula cannot be used to find  $x$  as  $m=0$ )
- Parallel lines (there is no single point of intersection)
- When looking for touching lines, some line sections are so short that both ends are within the circle, therefore, no intersections are found.

#### **4.2.2.1.2.2 Optimisation**

The processing is speeded up using bounding boxes for each line section, so that detailed processing is only performed on sections which are close enough for there to be a feature.

Processing time is also halved by only checking the line with unchecked lines. For example, there is no point checking if the fifth line intersects with second line if we have already established that the second line has no intersections.

#### **4.2.2.1.2.3 Self Intersect**

A separate step is performed to check whether the line intersects with itself. This happens frequently when dealing with circles.

#### **4.2.2.1.3 Corners**

*Corners* are found by marking points at which the drawing velocity drops to a certain threshold. Through experimentation it was found that the threshold could be set to zero without skipping any corners. This can be simply detected by looking at the previous point. If the position has not changed we know that the pen has stopped. This method can produce a lot of erroneous 'false' corners. These are removed in a later step (see Section 4.2.6.1).

#### 4.2.2.1.3.1 Switching off corner detection

For some shapes we know that there should not be any corners. For this reason an option was added to turn off the zero velocity detection.

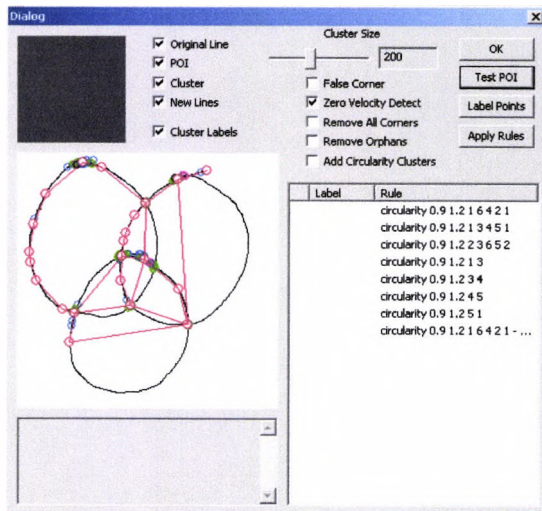


Figure 4.7 - Zero Velocity Detection Enabled

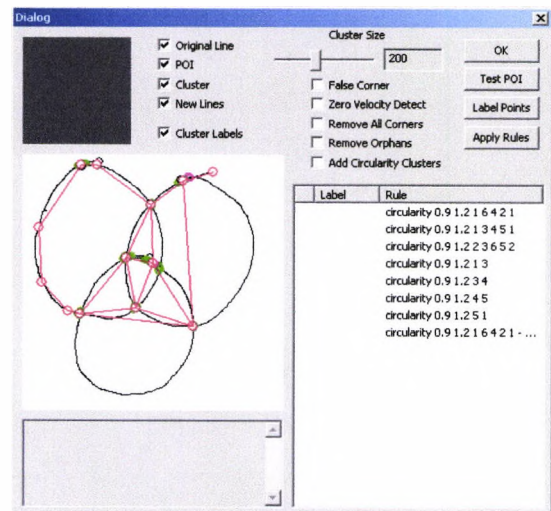


Figure 4.8 - Zero Velocity Detection Disabled

The stopping points on the circle were found to create a significant number of extra clusters that were not necessary. Figure 4.7 shows the clusters identified with zero velocity detection enabled. Figure 4.8 shows the cluster identified for the same drawing with the zero velocity detection disabled. The resulting drawing is a lot simpler. Extra points slow down the template matching stage of the analysis, so the performance of the system is improved if these points are not created.

### 4.2.2.2 Populating the list

The initial points of interest are stored as a list (C++ Vector type) of type `PossiblePOI`. This contains a pointer to the line element data and an integer containing the line number. This list is cleared and then the program scans through all the lines and creates a new element in the list for each line element that has been marked.

Figure 4.9 shows the class diagram after the Possible POI list has been populated.

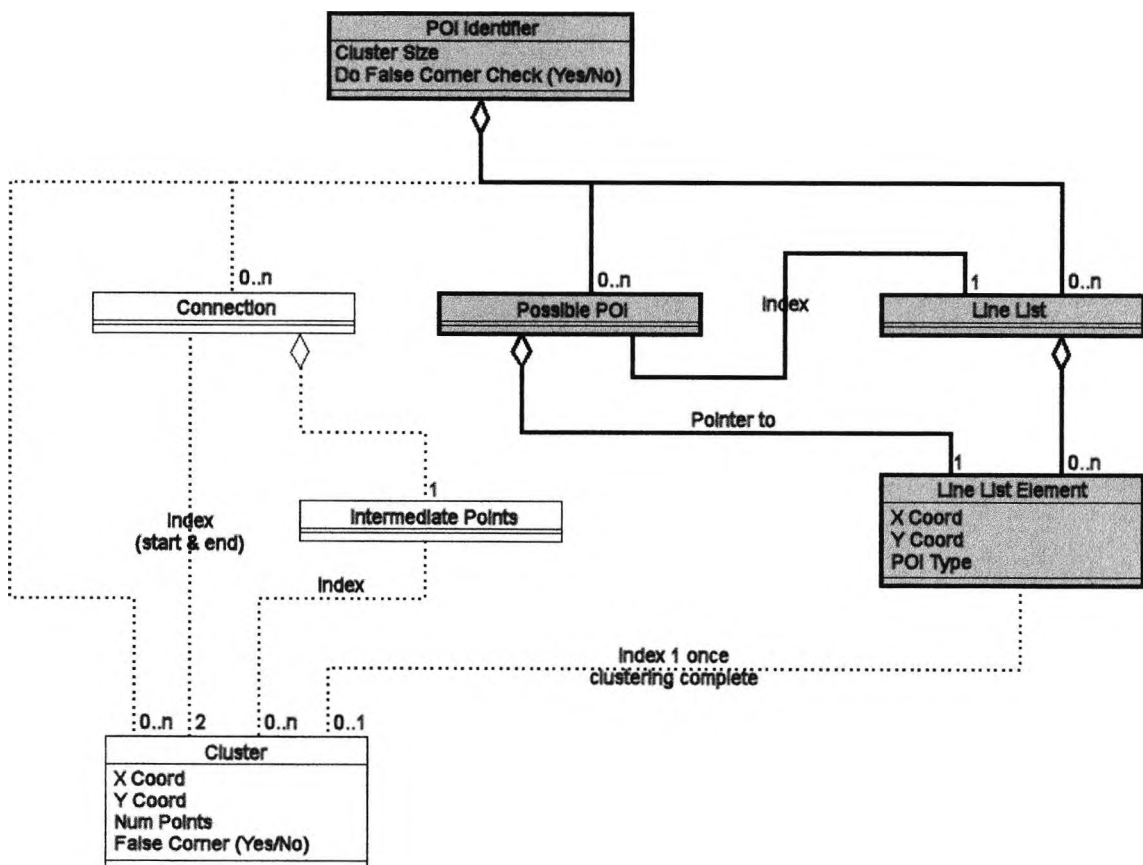


Figure 4.9 – Initial Interest Points

### 4.2.3 Clustering

Each feature of the shape needs to be represented by one point. The initial POI detection will almost always find more than one point at each feature. When two lines intersect, for example, there will be at least two points; one on each line. Clustering is applied to reduce the set of points to match the number of features.

Points are deemed to be within the same cluster if they are within a set Euclidean distance of each other. This distance is specified in the template as the cluster size. It can, therefore, be different for each shape. The user interface for the scoring component allows the cluster size to be changed; this enables the best value for each shape to be found experimentally.

Clustering is performed by applying the following steps to each POI in the list.

- First we assign it a new cluster number.
- Then we check if it is close to any other POI (within cluster size).
- Any POI that is close enough will be merged into the new cluster by changing its cluster number. If this POI is already in a cluster, all other members of the cluster must also be merged into the new one.

Once every POI has been incorporated into a cluster, the clusters are sequentially numbered starting from zero. Figure 4.10 shows the changes to the class diagram. The `LineListElement` now holds an index into the Cluster list. This list, however, has not yet been populated, so the next step is to produce the list of clusters.

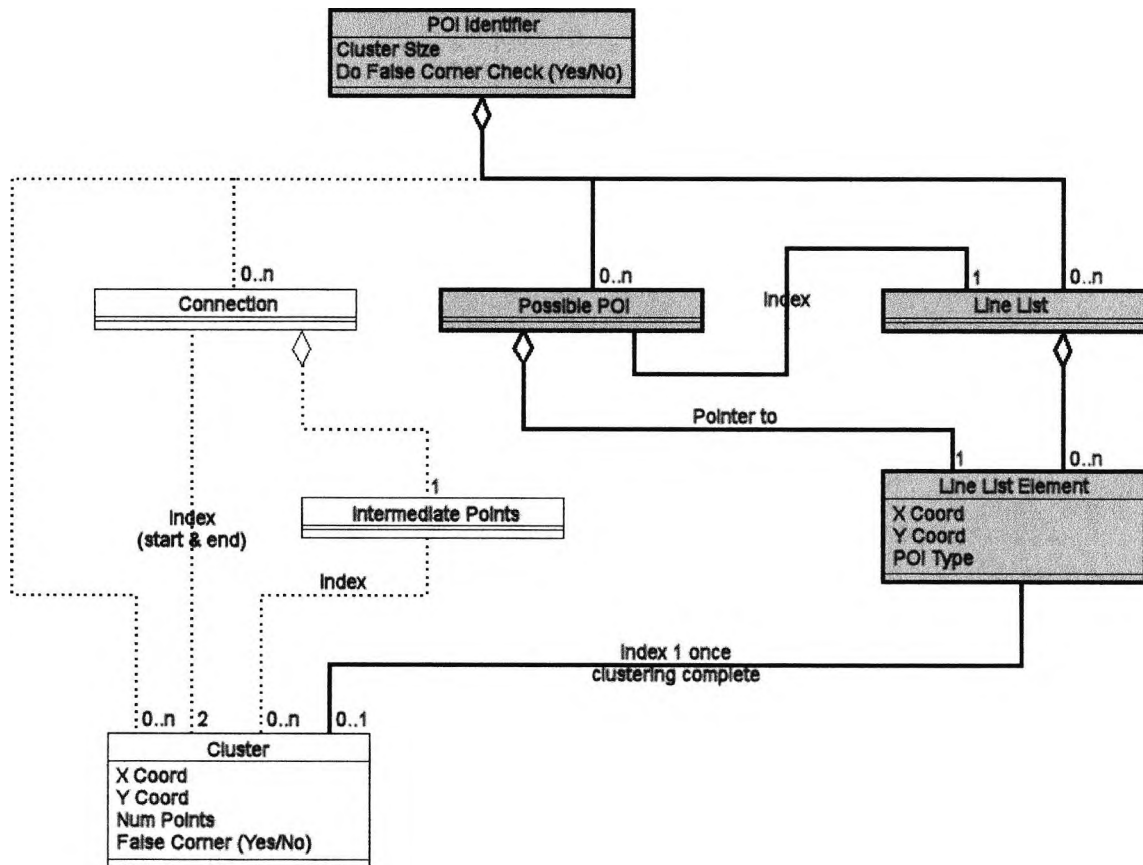


Figure 4.10 - Beginning Clustering

Each cluster is built up by scanning through the Possible POI list. If the cluster number matches we recalculate the centre point of the cluster. A count of the number of points is kept as it is required for the re-centring algorithm. Figure 4.11 shows the class diagram with the cluster list populated.

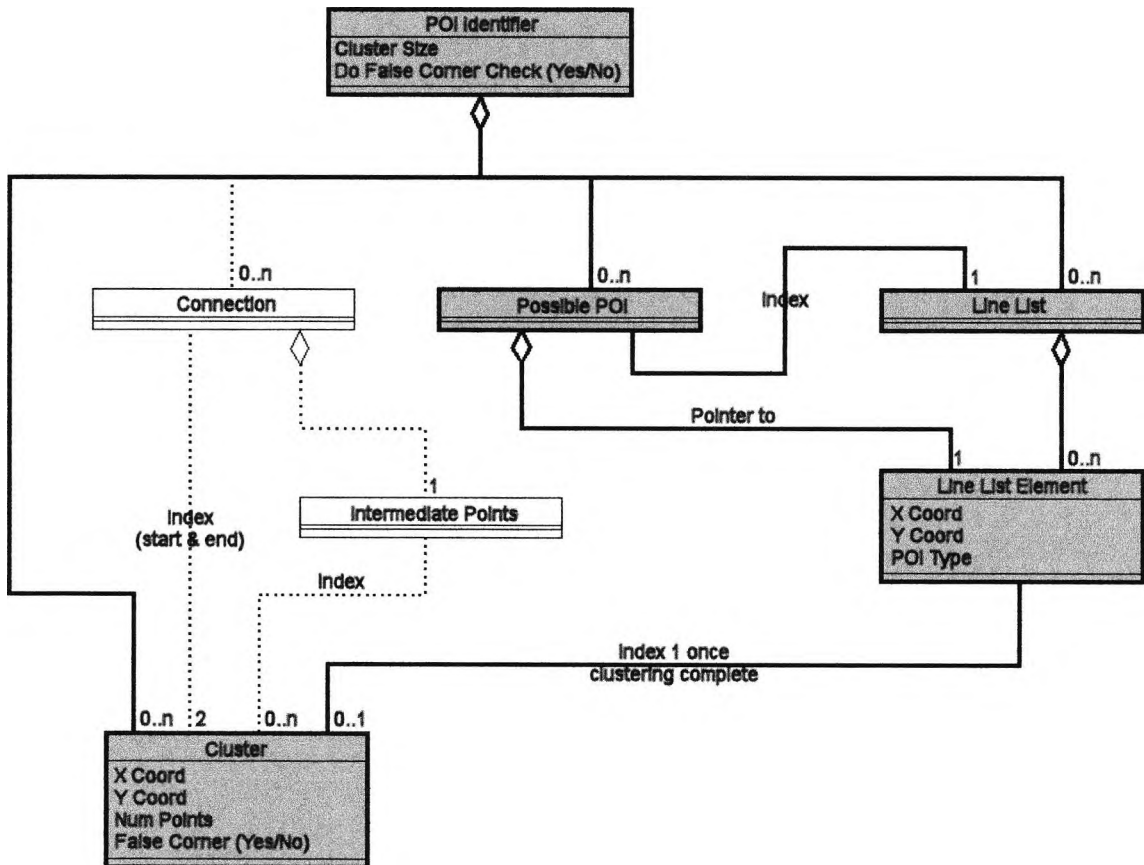


Figure 4.11 - Clustering Complete

#### 4.2.4 Finding Connections

Now that we have clusters marking the locations of the features, we need to find how they are linked together.

For each line we scan the elements looking for clusters. When one is found we store the cluster number as the start cluster. We then continue scanning until we find another cluster. When one is found we store the cluster number as the end cluster. We know that there is a connection between the start cluster and end cluster so we create an element in the connection list. We then set the start cluster to the current end cluster and continue to scan. This process is repeated until we reach the end of the line.

It is possible that two clusters may be connected to each other by more than one path, or by more than one line, therefore, once all the lines have been scanned, the list of connections is then checked for duplicated entries.

Figure 4.12 shows the class diagram with the connection list populated.

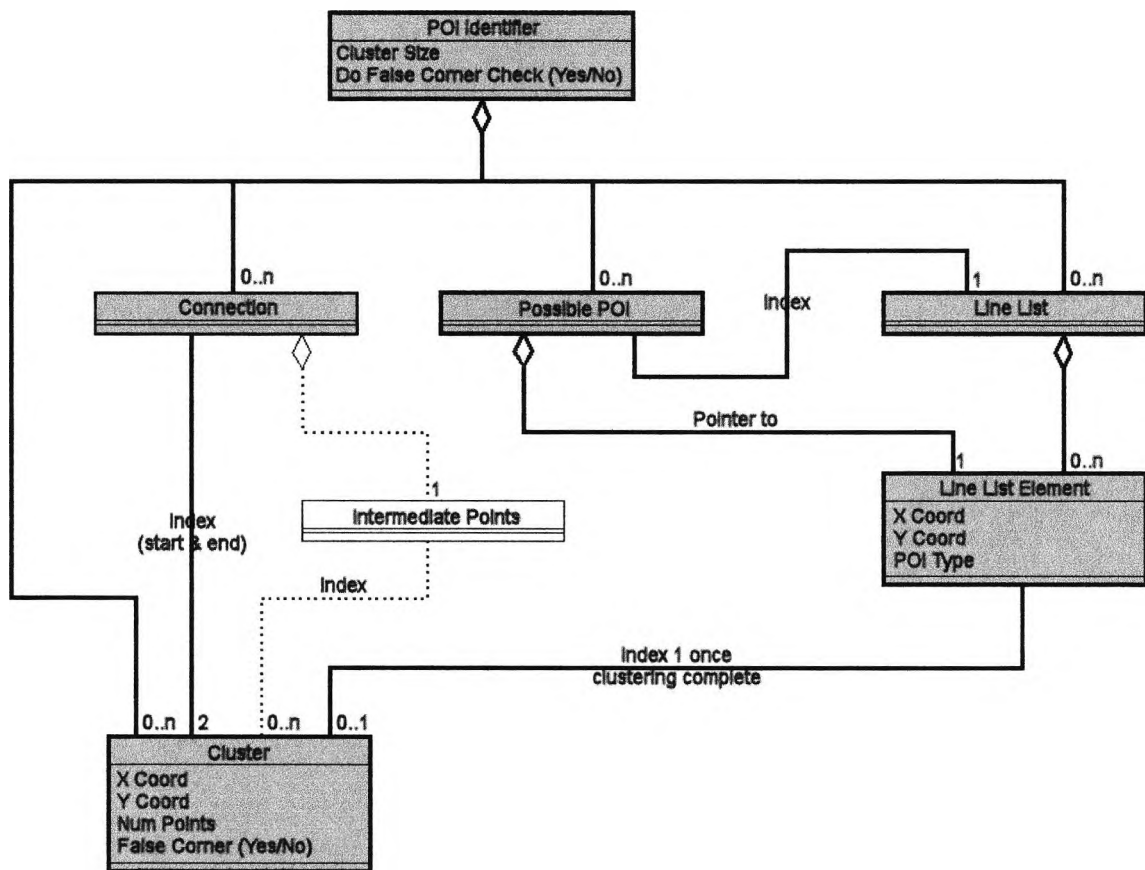


Figure 4.12 - Connections



### 4.2.5 Circles

To identify circles we need a method of specifying them in the template. If we only have one point on the circle we could specify “Point x is a circle” but there could be several lines through x and hence it is impossible to define which line should be the circle.

There are usually several clusters around the circle circumference. The pen stopping, the circle being constructed from more than one line, lines touching the circle and the start and end points of the line all create clusters. By specifying more than one point we can make it clear which path the circle should take.

Two points makes the circle identifiable, although false matches would be produced as we would not be able to distinguish between a single line and a circle. If there is a route from A to B there will always be a route from B to A.

With three points we can specify the path round the circle easily. Cluster A must connect to B which must connect to C which must in turn connect back to A.

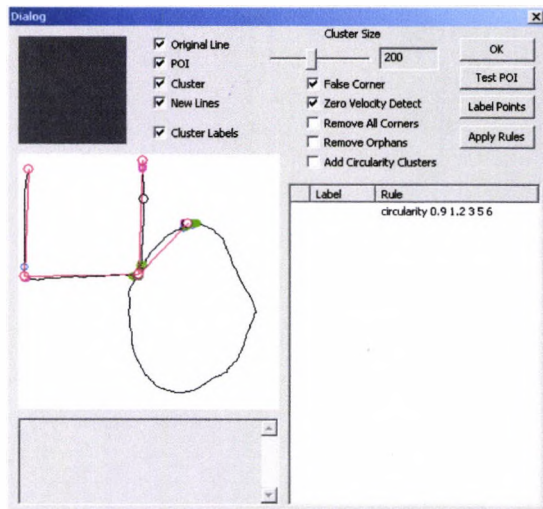


Figure 4.13 - Only two points found on circle

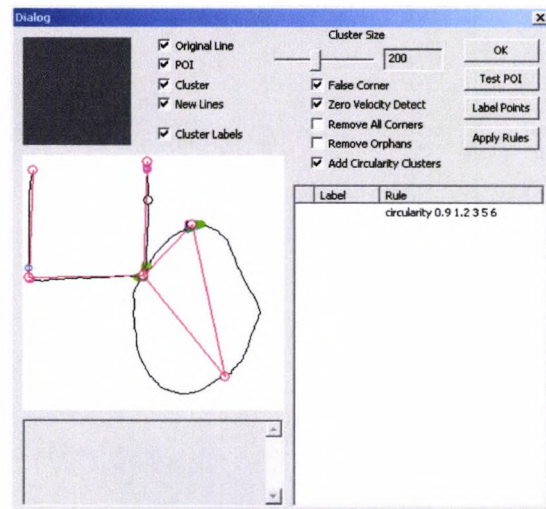


Figure 4.14 - Extra point added

When we are creating the cluster connection list (see Section 4.2.4) we keep a look out for duplicate cluster links. Figure 4.13 shows a shape in which the circle has only 2 clusters. In this case there are two different ways of getting between the points,

namely clockwise and anti-clockwise. Instead of discarding the duplicate clusters, as previously done, we now create an extra cluster in the middle of line which makes the duplicate connection. This will turn a two cluster circle into a three cluster circle as shown in Figure 4.14.

While we are scanning the lines for connections we can also look out for lines with no connections. These must be either circular or so short that both ends are within the same cluster. When we find such a line we search through to find the biggest gap between POI elements. Then we add two extra clusters, equally spaced, into this section of the line. This turns a one cluster circle into a three cluster circle as shown in Figure 4.16.

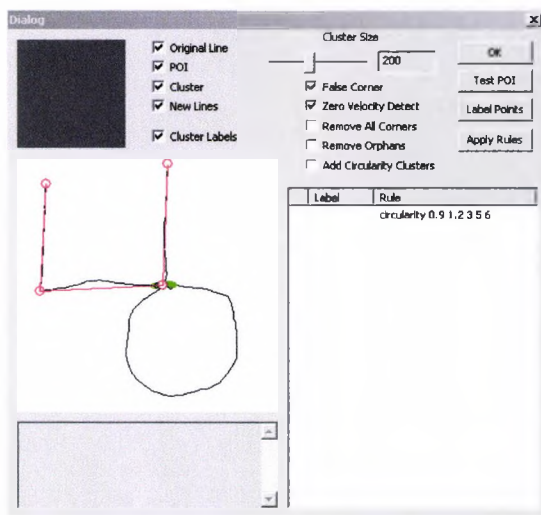


Figure 4.15 - Only one point found on circle

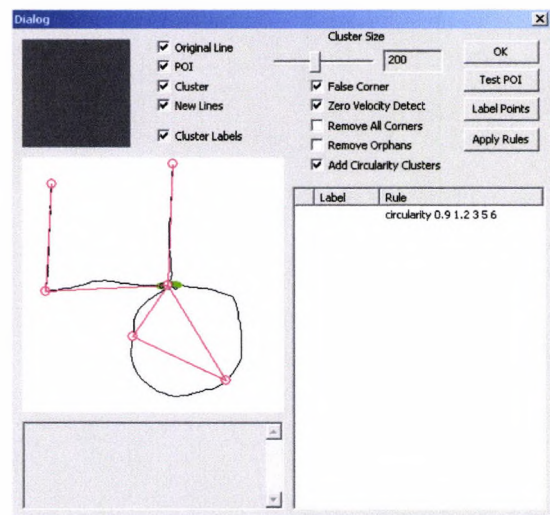


Figure 4.16 - Two extra points added

There is an option to turn this off if we know there are not any circles in the shape we are assessing.

### 4.2.6 POI removal

Some points may be drawing artefacts rather than features of the shape, so these clusters are removed. The types of artefacts are described in the sections below. Removal of these extra clusters completes the POI identification section of the scoring process.

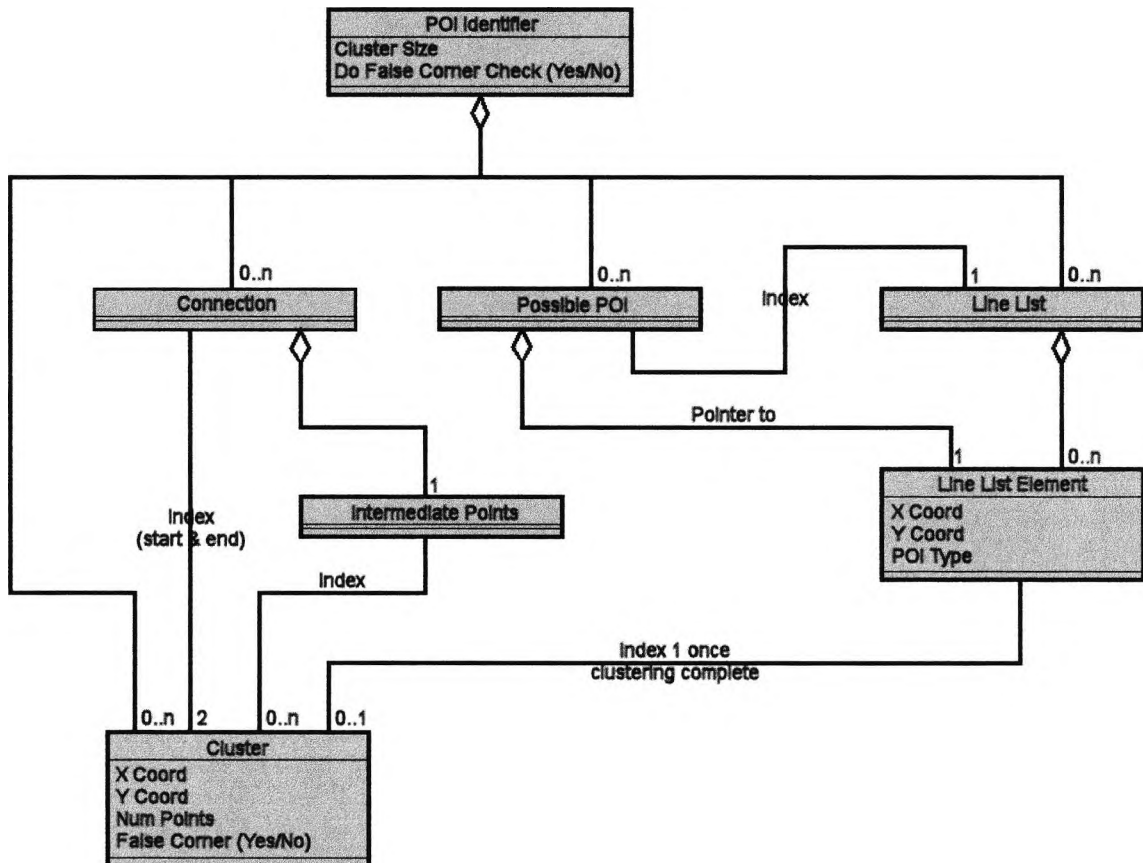


Figure 4.17 - POI Identification Complete

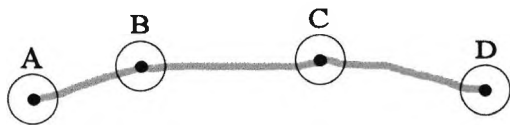
#### 4.2.6.1 False Corner Detection

One problem with using a zero velocity to identify corners is that these can occur during straight sections of a line. Also if a line has been extended there will be a cluster at the join which does not affect the shape of the drawing. An algorithm was developed to reduce the effect of this.

First we have to determine if the cluster could be a corner point. This is done by checking how many other clusters it is linked to. If it is linked to exactly two other clusters then it is either a corner, a line continuation or a stopping point along a line. We keep note of the two clusters it is linked to.

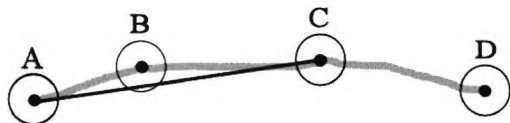
Once this has been done we check the linearity of the line going through all three clusters. This is done using the cluster size as a guide. If the cluster at the centre point is intersected by a straight line drawn between the two other clusters we can say that it is a false corner. It is then marked for removal.

We find all of the false corners before removing them to deal with the situation where there are two or more false corners along the same line. Figure 4.18 shows a situation where removing the false corners as they are detected would miss out a cluster which should be removed.



**Figure 4.18 - Line with a false corner**

Clusters B and C are continuations of the same line rather than corners. We first test cluster B.



**Figure 4.19 - First erroneous cluster detected**

Figure 4.19 shows that the line between the centres of clusters A and C passes through the cluster boundary of cluster B. This is our criteria for removing the cluster. If this is done immediately, cluster C will now be between clusters A and D. When this test is performed the cluster is not marked for deletion, as show in Figure 4.20.

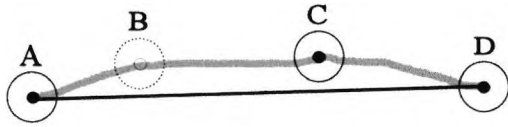


Figure 4.20 - Second erroneous cluster missed

Cluster C is not marked as a false corner. By finding all the clusters first before removing them we test for C being between clusters B and D as shown in Figure 4.21.

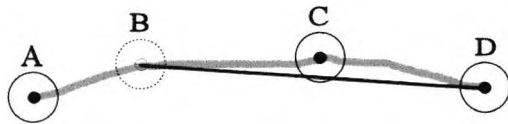


Figure 4.21 - Second erroneous cluster detected

In this case cluster C will be successfully marked as a false corner. Once all the potential corners have been checked, both B and C can be removed.

Removing the false corner is not entirely straightforward. The cluster is flagged as being a false corner in the list of clusters. The list of connections must then be updated. We search the list for the two connections and combine them to make one new connection.

For example

1->2 then 2->3

becomes

1->3

The connections could be listed either way round in the list so there are four possible cases to allow for, namely:

1->2 then 2->3

1->2 then 3->2

2->1 then 2->3

2->1 then 3->2

We then have to scan through all of the lines to find all elements from the false corner cluster. The POI type of these elements is then changed to flag them as being a false corner.

### 4.2.6.2 All corner removal and orphan removal

Some of the shapes have no corners. For example all of the expected clusters in shape 16 have 4 connections from them. This shape contains circles which are often drawn with several lines. Thus, the drawing will have a number of false corners. Also as they are on a circle they may not be removed by the normal algorithm. By extending the false corner detection to remove all corners we would solve this problem. Figure 4.22 shows a drawing of shape 16 with all corner clusters removed.

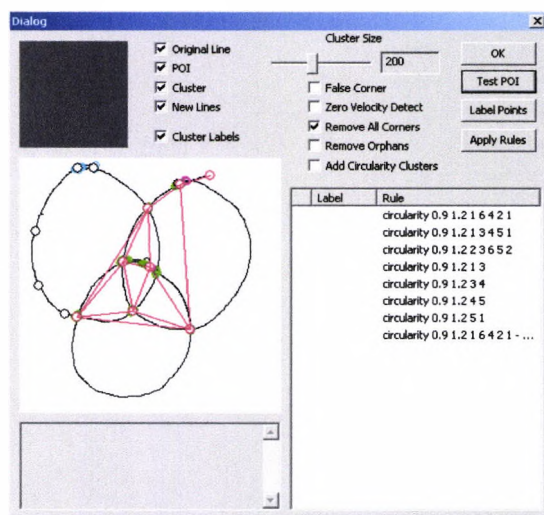


Figure 4.22 - All two connection clusters removed

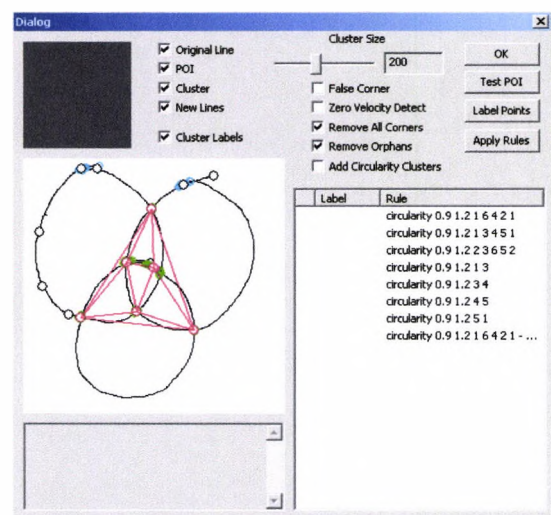


Figure 4.23 - All one and two connection clusters removed

There is one more problem with circles which needs to be addressed. Sometimes, where the circle joins up on itself, there may be a 'tail' coming off the side of the circle. This manifests itself as a cluster with three connections and a cluster with only one connection, as can be seen in Figure 4.22 at the top of the top right circle. This was solved by removing all clusters with only one connection, then removing all clusters with two connections as show in Figure 4.23.

The drawing can now be matched to the template as shown in Figure 4.24. The template matching process is described in the next section.

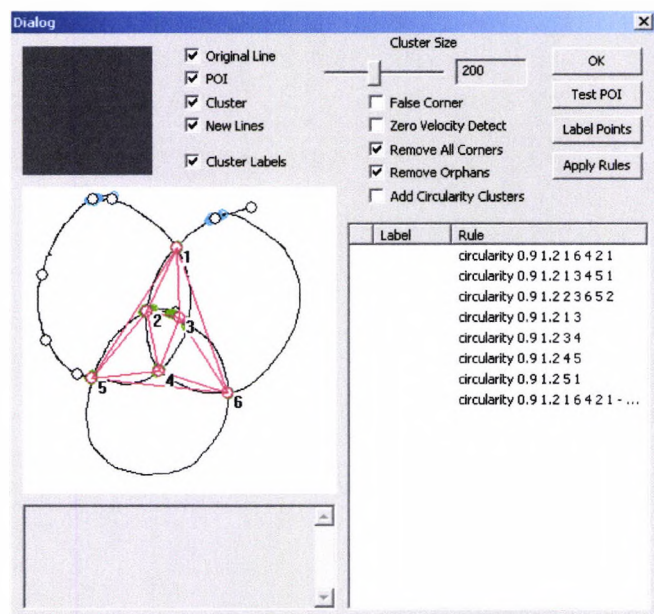


Figure 4.24 - Matching to template

### 4.3 Template Matching

The purpose of the template matching stage of the algorithm is to label the features of the child's drawing. The actual lines drawn are not used for this process, only the clusters and connections that were detected in the previous stage.

The template is a list of rules defining how points relate to each other. Each rule specifies two points and the relationship between them. The template can specify connections and also direction. For example:

Point 1 connects Point 2.

Point 3 is above Point 6.

Figure 4.25 shows one of the shapes as illustrated in the VMI manual. A template is produced based on this ideal shape. We have numbered each of the points where we expect a cluster to be found in a child's copy of this shape. These numbers are used to specify the rules that make up the template for this shape.

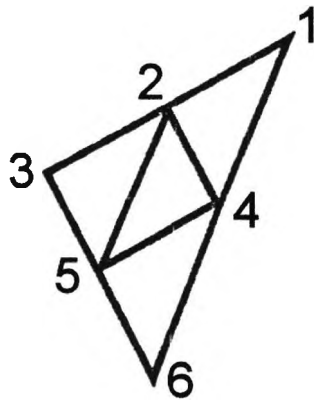


Figure 4.25 - Numbered Points

The number of rules that need to be defined depends on the shape. There needs to be enough rules that there can only be one way of numbering the points whilst still obeying them all. For example, in the shape shown in Figure 4.25, if we created rules just for the connections, there would be three ways of numbering the points. This is because the shape could be rotated without breaking any of the rules. Extra rules would be needed to ensure the orientation of the points is correct.

The domain of possible solutions can be searched to find which clusters from a child's drawing match each point in the template. This search has been optimised to reduce the time taken to find solution.

If the numbering is unsuccessful then we cannot match the drawing to the template. This may be due to errors in the software or because the drawing has not met the VMI criteria. In either case we cannot continue the scoring and must mark the drawing as a fail.



### 4.3.1 Template

The template consists of a set of rules which describe the ideal shape. Each rule specifies a condition that must be true for the template to be correct. The rule consists of the condition itself and two points which it refers to.

Each point in the template needs to be matched to a cluster. Using just this simple rule set, it is possible to create a template which will always match the appropriate clusters to the same points. If the drawn shape is incorrect or the POI identification fails then the template may not be able to produce any matches. With too few rules or incorrect rules the template may be able to produce more than one match.

A separate program was written to help design the template. Figure 4.26 shows the user interface for this program.

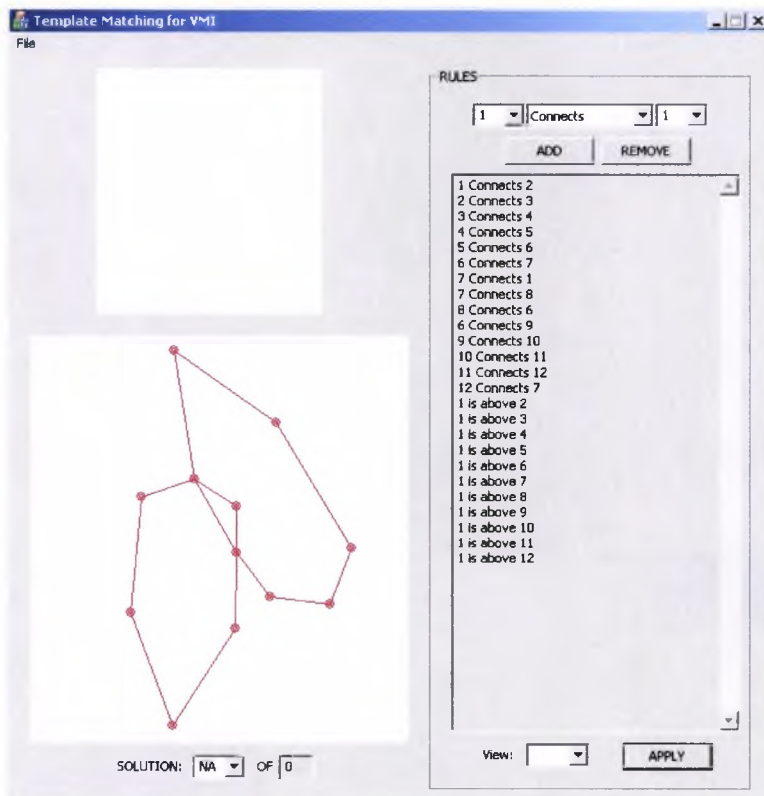


Figure 4.26 - Creating Template

The user can create a skeleton of the ideal shape by using the mouse. Double clicking adds a new point. By clicking on a point and holding the mouse button the points can be dragged to a new location. The right mouse button brings up a menu which allows points or connections to be created and deleted.

The template can be built up by adding and removing rules from the list. The list box on the right hand side of Figure 4.26 shows the set of rules defined for the shape.

By clicking the button labelled 'apply' the software searches for matches. The points are then labelled according to the first match found (see Figure 4.27).

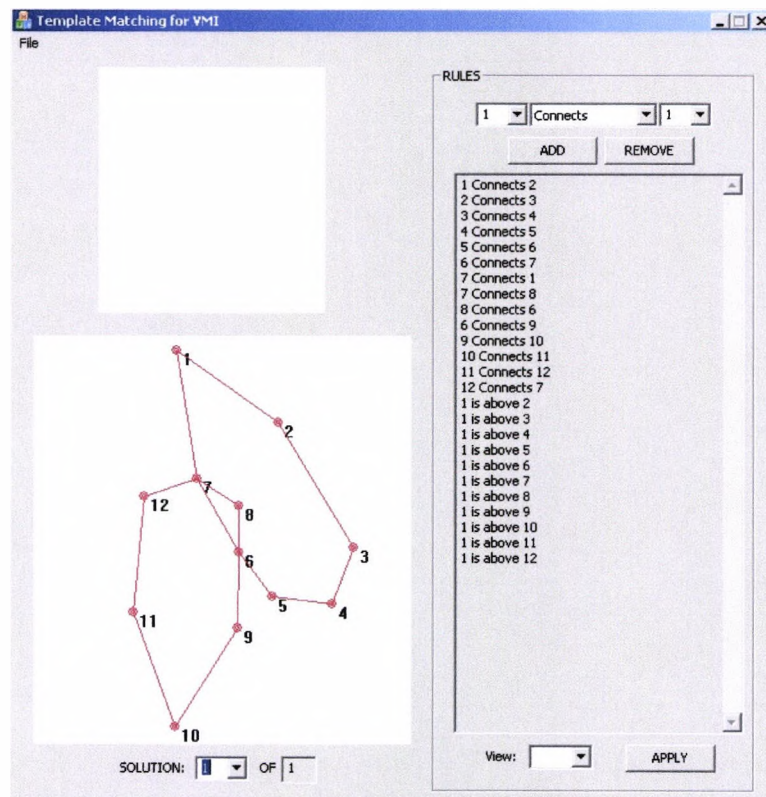


Figure 4.27 - Template match found

If more than one match is found the user can select which solution to view from the drop down box below the skeleton shape (see Figure 4.27). In this case more rules need to be added to eliminate the erroneous matches.

### 4.3.1.1 Template Rules

There are four possible conditions which can be imposed by the template, which may be characterised as follows:

- Connects
- Connects Indirectly
- Is Above
- Is to the Left of

#### 4.3.1.1.1 *Connects*

There is a line connecting the first point with the second point. This is easily tested as the POI Identifier creates a list of all the connections.

#### 4.3.1.1.2 *Positional*

There are two positional rules, namely:

- Above
- To the Left of

If we want to specify that point A is below point B we can use the *Above* rule and swap the points round. The rule for this would be: B Above A.

Similarly, we do not need to specify a rule for: To the Right of.

The x and y coordinates of each of the points are known, so performing the test is simple.

#### **4.3.1.1.3 Indirect Connection**

This rule was incorporated specifically for dealing with circles. There may be many clusters around a circle. Unless there were exactly three we would be unable to find a match using the normal connection rule. "Indirect Connection" specifies that two clusters are linked but there may be other clusters in between. Only clusters that have not been matched to the template can be traversed in the indirect connection, this is to stop the search simply going back the same way round the circle.

To do this we search all of the possible connections from our current point. At each point we come to we recursively search again. The search stops (for that recursion) when we reach our target, or we hit a point we have already visited, or we hit a point that has already been labelled by the template matching. We need to search in a tree-like fashion as there may be spurs coming off the circle where the lines meet up.

To specify a circle in the template we use three points making a triangle. One side will be an indirect connection. The other two sides will be linked with the *Connects* rule.

### 4.3.2 Implementation

Figure 4.28 shows the class diagram for the template matching process. The diagram only needs to be shown once as this process is performed in one step.

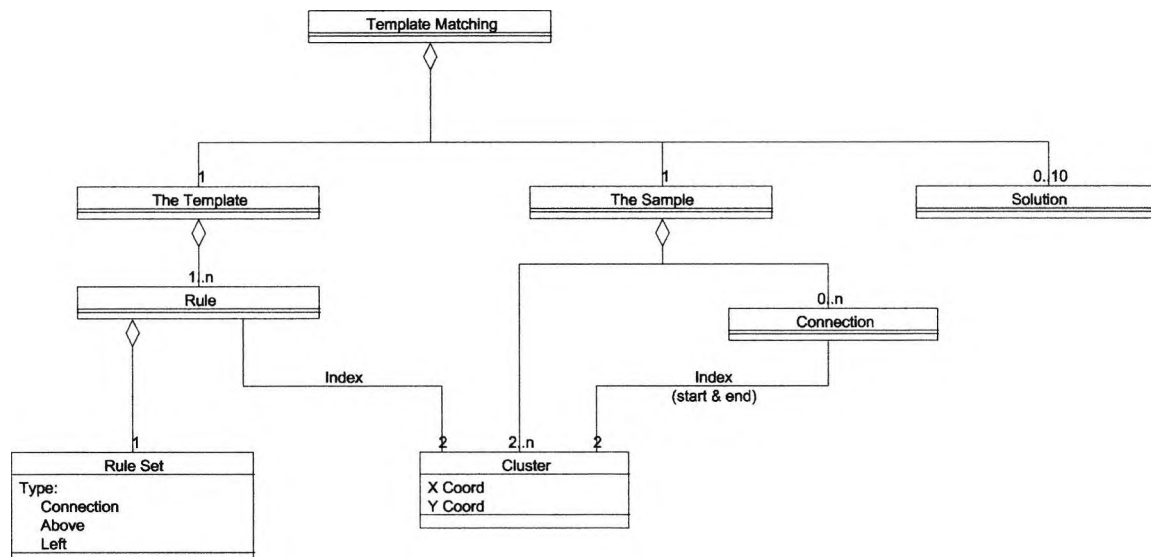


Figure 4.28 - Class Diagram for Template Matching Stage

There are three main data classes, as shown in Table 3.

The Template	This contains the template. A set of $n$ rules which consist of a rule type taken from the Rule Set and two indexes to Clusters.
The Sample	The sample is the drawing data that needs to be labelled. The Connection and Cluster information comes from the POI identification step.
Solution	This stores the solution maps

Table 3 - Template matching classes

All classes except for the solution are populated before the processing begins.

### 4.3.3 Search algorithm

Initially an exhaustive search was performed by stepping through all the possible combinations of cluster points and template points. This was very time consuming. For example: shape 27 of the VMI test has 36 points. An exhaustive search would provide  $3.7 \times 10^{41}$  possible solutions.

The search algorithm was optimised extensively to create a feasible method of analysis. This will be explained by the use of an example. To prevent the diagrams being too large a simple shape will be used. This is not a shape from the VMI test, but illustrates the principle easily. Figure 4.29 shows the example shape with the points labelled. Letters have been used for the labels instead of numbers to distinguish them from the cluster numbers from the child's drawing.

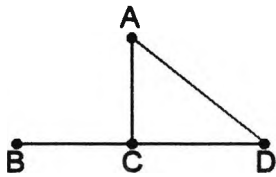


Figure 4.29 – Simple Example Shape

The template rules for this shape are:

- A Connects C
- A Connects D
- B Connects C
- C Connects D
- A Above D

Just using connection rules it would be possible to swap A and D. Therefore adding a positional rule ensures there is only one match to the template. We only need to add one positional rule; 'A is left of D' would be equally appropriate.

An example of a child's drawing is show in Figure 4.30. The numbers are at the points where clusters would have been found during the POI identification stage. The numbers would also have been assigned during the POI identification process.

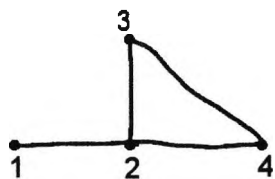


Figure 4.30 - Example Child's Drawing

Figure 4.31 shows the search tree diagram for an exhaustive search. The set of template labels is tested against every possible combination of clusters in the drawing. In an exhaustive search we only look at the leaves of the tree. We therefore test the complete set of rules 24 times.

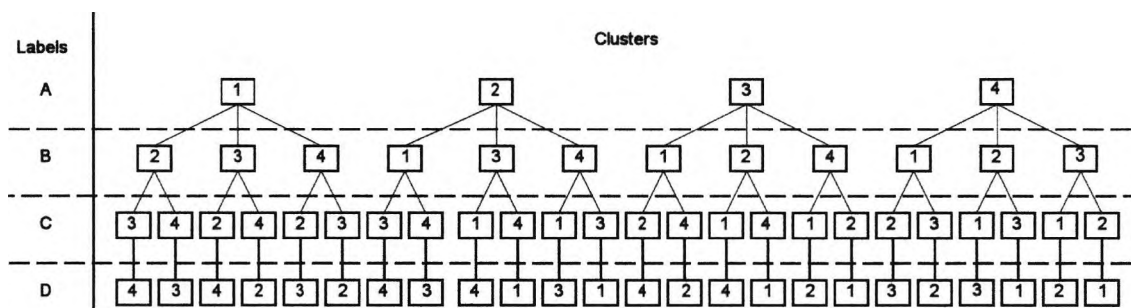


Figure 4.31 – Exhaustive Search Tree

By counting the number of connections at each point in both the template and sample data we can eliminate many of our search branches. The leaves of the search tree will only be reached if the proposed solution matches the number of connections at every point. Only then does the complete set of rules need to be applied.

Table 4 and Table 5 show the number of connections on the template and the child's drawing respectively. Note that the template labels will always be the same whereas the cluster numbers will change for different drawings.

Label	Connections
A	2
B	1
C	3
D	2

Table 4 - Template

Cluster	Connections
1	1
2	3
3	2
4	2

Table 5 - Child's Drawing

Figure 4.32 shows the branches in the search tree which we can now eliminate from the search. We can see that Label A has 2 connections. Cluster 1 has only 1 connection so everything below this node can be removed from the search. The same applies for cluster 2. Only cluster 1 will match label B so 2/3s of the remaining branches can be removed at the next level. Nodes which are greyed in the diagram need not be processed.

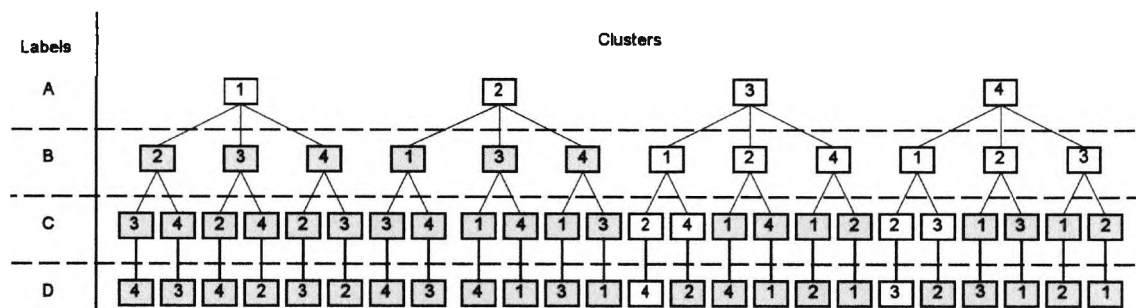


Figure 4.32 - Initial Optimization

In this method we perform a connection count comparison 14 times and apply the full set of rules twice.

Comparing the number of connections adds an extra step to the processing. However, it is a very quick process. We are not comparing what the connections are, just how many are present. Counting the number of connections is done only once, before the search begins. The comparison is merely checking if two table entries are equal.



This was further optimised by reordering the search so that the points that are least likely to be found are looked for first. We group the points of the template by the number of connections they have. The groups with the fewest members are placed at the root of the tree and so on until the biggest group is at the leaves.

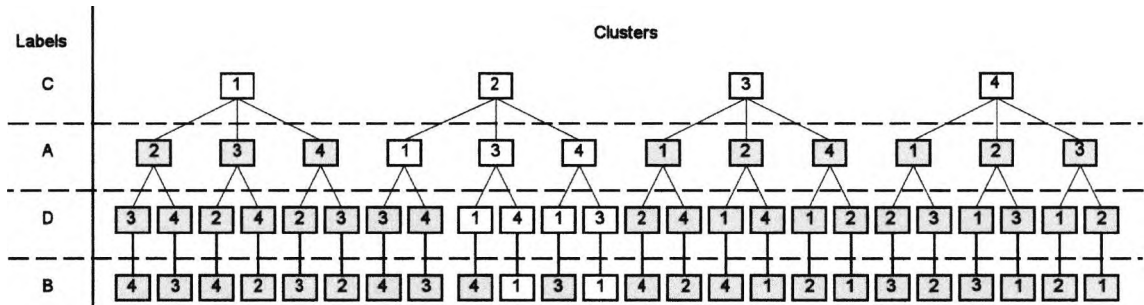


Figure 4.33 – Final Optimization

In this method we perform 11 connection count comparisons and apply the full set of rules twice.

Optimization	Connection Count	Test Rules
Exhaustive Search	0	24
Initial Optimization	14	2
Final Optimization	11	2

Table 6 - Comparison of search methods

Table 6 shows how the performance of the algorithm was successively improved with each optimisation. In this simple example we can see that the optimized search algorithm reduces the number of tests dramatically.

## **4.4 Absolute Rule Framework**

The absolute rule section of the process ensures that the drawn shape meets all the criteria for scoring.

The rules can look at the positioning and connectivity of the clusters as well as looking at the actual drawn lines.

The absolute rules are very specific to a set of shapes and scoring criteria. In order to score a test other than the VMI a different set of rules would need to be implemented.

For this reason the software for this process was developed in two stages.

- A general framework which could be used for any set of shapes.
- A set of rules which are specific to the VMI test

In this chapter we describe the framework which allows the scoring to take place. The implementation of the individual rules required to score the VMI test are described in the next chapter (see chapter 5).

### **4.4.1 Template**

The rules are specified as text commands. Each rule has a name which is specified followed by a set of parameters.

Figure 4.34, shows the text script for the absolute rule section of shape 11 of the VMI. Blank lines are ignored, as are lines starting with //. This enables the script writer to add comments that the user will not see. The command 'Comment' is a rule which always passes and simply echoes the preceding text to the screen. This is used to add comments which the user can see.

```
// -- Absolute Rule Section --

// Rule 1: Two Intersecting Lines
Comment Rule 1
TwoInt 1 2 3 4 5
Comment

// Rule 2: Lines angles between 20-70 degrees and 110-160 degrees
Comment Rule 2
AbsAng 3 1 20 70
AbsAng 4 5 110 160
Comment

// Rule 3: Longest of 4 unextended legs no more than twice a long
// as shortest (not including extensions)
Comment Rule 3
CompLeg 0 2 2 1 3 4 5

// Test
Comment
Comment Invalid Command
AbsAng 3 1

// -- End Absolute Rule Section --
```

**Figure 4.34 - Script for shape 11 of the VMI**

Figure 4.35 and Figure 4.36 show how this script appears in the Scorer program. By highlighting the rules the user can see details of the calculation. The AbsAng rule shown in Figure 4.35 calculates the angle between two clusters and compares it to limits specified in the parameters. The angle calculated is shown when this rule is highlighted. Also the line between the clusters used is shown on the diagram. In Figure 4.36 the rule uses the actual drawn lines to perform the calculation. The straight lines between the clusters specified are shown in pink, and the drawn lines are highlighted in green.

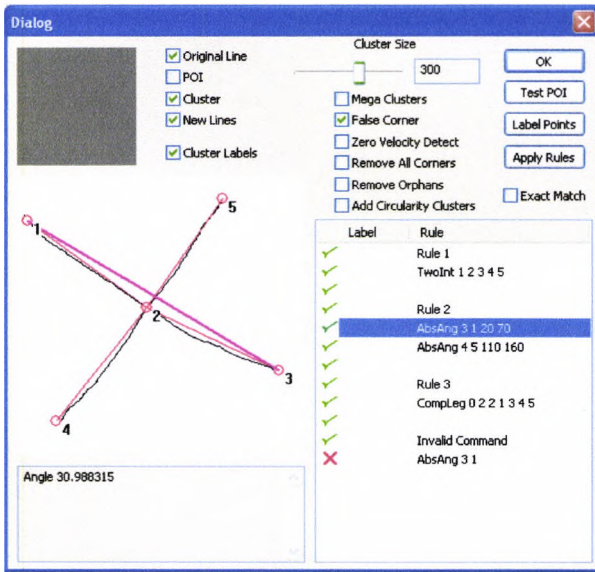


Figure 4.35 - AbsAng Display

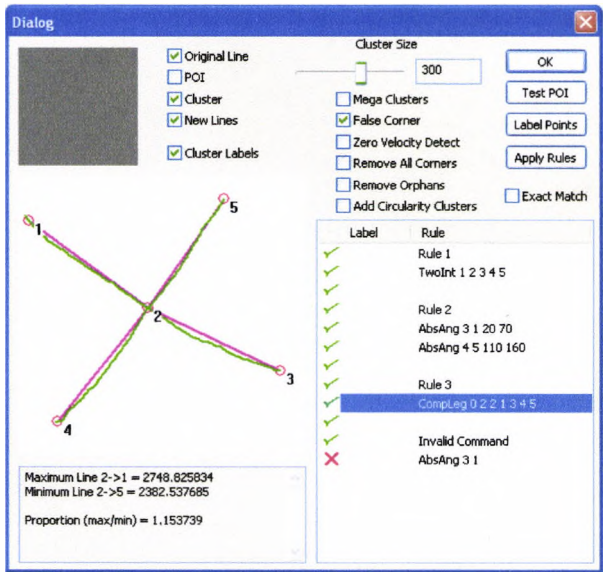


Figure 4.36 - ComplLeg Display

If the parameters for the command are incorrect a description of the correct syntax is displayed. Figure 4.37 shows what is displayed when an invalid command is found in the script. An error message is generated, followed by a description of the correct syntax for the command.

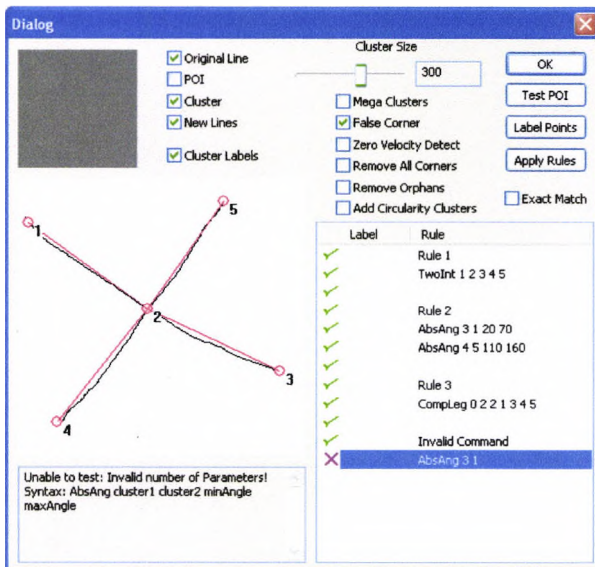


Figure 4.37 - Invalid Command

### 4.4.1.1 Higher Commands

For certain shapes it was necessary to introduce higher level commands. These are commands which affect the outcome of other commands. Some shapes in the VMI specify that one of a set of criteria must be matched. For example, in shape 17 (see Figure 4.38) one of the criteria is 'Baseline and at least one other side straight'. We can easily test all three sides for straightness but we need some way to allow one of the diagonal sides to fail the test without causing the whole shape to fail.

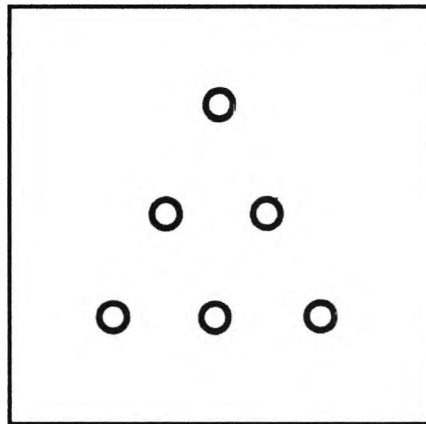


Figure 4.38- VMI shape 17

This was solved by allowing the rules to be labelled. Labels are placed before the rule name in the script. The label can be any single word (no spaces) and is preceded by the word 'Label'.

For example, in the template for shape 17 we use the rule `SideInt` (see Section 5.4.12) to test the straightness of the sides. To add the label 'LeftSide' we would write the rule as:

```
LabelLeftSide SideInt <parameters>
```

To allow a rule to fail we create a higher level command. The most basic is the `MustPass` command (see Section 5.4.14) which takes the number of passes required as the first parameter and a list of command labels as the remaining parameters.

The full script for the rule: Baseline and at least one other side straight

```
SideInt 1 4 6 5
LabelLeftSide SideInt 1 1 4 2
LabelRightSide SideInt 1 1 6 3
MustPass 1 LeftSide RightSide
```

If a rule fails but is allowed by a higher rule then its status is changed to a third state to show clearly that it has failed but is allowed through. After scoring it is labelled as accepted (designated “OK”) rather than fail.

Figure 4.39 shows how the labels are shown on the Scorer display. In this example we can see that the rule labelled ‘RightSide’ has failed, but because of the higher rule this failure has been tolerated, it is marked as ‘OK’.

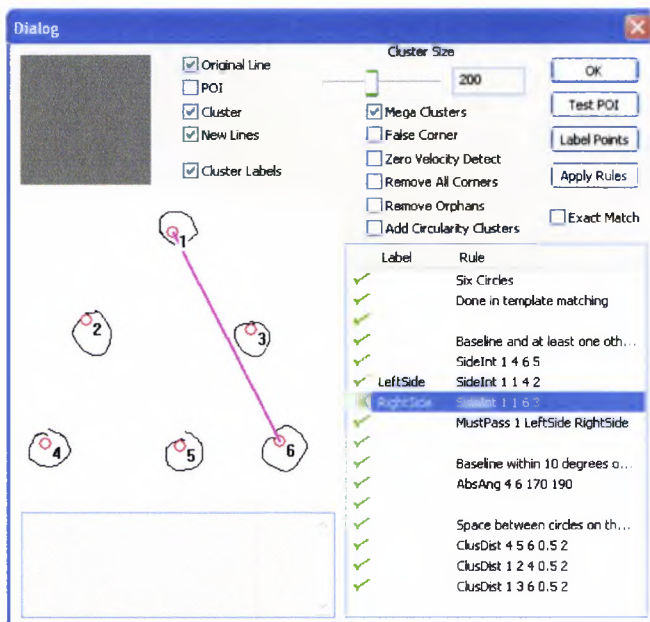


Figure 4.39 - Higher Rules

### 4.4.2 Implementation

For each rule that we specify in the script we must implement its functionality in the scorer software. Each rule is implemented as a separate class. All rule classes inherit a degree of functionality from a super class. The super class is called ‘ARule’.

The list of rules is stored as a Vector of `ARule` objects. This is a polymorphic list so each object may be of any type sub-classed from `ARule`. This enables us to simply step through the list and call the ‘`applyTest`’ function of all of the rules without needing to know what exact type of rule each object is.

Because we have higher rules, which can change the outcome of previous rules, we store the results in an array. This also makes displaying the scoring progress easier.

The array is implemented as an array of integers, and each element may take one of 4 values:

- -1: Not yet scored
- 0: Fail
- 1: Pass
- 2: OK (Fail but allowed by higher function)

In order to allow higher functions to operate, the results array is passed to the ‘`applyTest`’ function of the rule objects.

### 4.4.2.1 Adding a new rule

In order to create a new rule we must first define a new class. This class must have `ARule` as a super class. This new class has a function ‘`applyTest`’ which implements the scoring procedure for this rule, returning a true or false value for pass or fail respectively.

This new class must be linked into the rest of the system by modifying the `Template.cpp` file. There are two simple changes that need to be made.

At the top of the `Template.cpp` file there is a list of `#include` statements to link all of the rule classes. A `#include` for the new class’ header file must be added to this list.

Also in the `Template.cpp` file there is a function ‘`loadAbsoluteRule`’. This function parses the script, creates the correct object for each rule and passes it the string of

parameters. The main section of this function is a big 'if else' statement. The rule classes contain a static member function, which will return a text string holding the name of the rule. The name of each rule is tested against the command specified in the script. If the names match, an object of that type is created. So that the program will recognise the name of the new rule, an extra 'else if' clause must be added to this function.

## **4.5 Conclusions**

We have developed a flexible scoring system which can be used to assess a variety of hand drawn shapes. This has been designed so that it can be easily incorporated into any Visual C++ software project. It can be run without a dialog screen, but also one can be opened up if necessary to view the details of the scoring.

A test harness has also been created which allows a database of templates and drawings to be easily assessed. This is able to share databases with the Software Assisted Scoring program described in Chapter 3.

The next chapter describes how the absolute rule section was extended to verify the formal rules specified in the VMI test, and how the system has been used to score the VMI test in practice.



# Chapter 5

## Scoring VMI using General Scorer

### **5.1 Introduction**

The previous chapter describes computer software which has been developed to assess a set of drawn shapes against a set of criteria (see Chapter 4). This chapter describes how the General Scoring software has been adapted to assess the Beery Developmental Test of Visual-Motor Integration (VMI).

A set of software algorithms, or rules, has been created which allow the entire set of VMI scoring criteria to be verified.

For each VMI test shape, a template was designed which contains all of the data necessary to perform the assessment, including:

- Parameters for feature extraction.
- A set of geometrical and positional rules to enable clusters labelling.
- A set of absolute rules to validate the shape.

The scoring performance of the software was assessed by comparing scoring decisions made by the software against results from human analysis.

## **5.2 Definition of Terms**

For the description of rules in this chapter we use some terms to mean specific elements of a drawing, specifically:

- Rule
- Command
- Cluster
- Point
- Point of Interest (POI)
- Line

These are defined in the following sections.

### **5.2.1 Rule**

A rule is a software implementation of a scoring criterion. This is a C++ class which is derived from `ARule` (see Section 4.4.2).

### **5.2.2 Command**

A command is a test string which is used to invoke a rule. This contains the name of the rule and a list of parameters which specify which parts of the drawing are to be measured.

### **5.2.3 Point**

A point refers to a sample point which was measured by the graphics tablet during the execution of the modified VMI test. A point has a position described by x and y coordinates.

### **5.2.4 Cluster**

Following feature analysis, points which are marked as having features are clustered together (see Section 4.2). At each of these ‘clusters’ the centre point is found. This centre point indicates the position of a feature of the drawing. The clusters are mapped to a numbering system which links each expected feature of the shape to a cluster on the drawing (see Section 4.3). After this mapping has been performed the cluster number, for any correct drawing, will always reference the same feature of the shape.

### **5.2.5 Point of Interest (POI)**

Following feature analysis, certain points are marked as having features (see Section 4.2). These points are termed ‘Points of Interest’ or POI. These points will be associated with a cluster. A POI will contain the following information:

- Position, stored as x and y coordinates.
- Feature Type (see Section 4.2)
- Cluster Number

### **5.2.6 Line**

A line is a set of points which were captured by the graphics tablet during the same pen stroke. A line starts when the pen touches the tablet and finishes when the pen leaves the tablet. The lines that make up a drawing are separated and are stored in an array. A single line can be accessed by using an index into this array.

## **5.3 Line Calculation Algorithms**

Several algorithms have been devised to compute measurements from, and locate positions on, the lines. These are used in several of the rules so are presented before we move on to the description of the rules.

### 5.3.1 Inner Line

Some rules can be applied by simply looking at the relative positions of clusters. Other rules require that we analyse the points that make up lines in the drawing. The most common operation we need to perform is finding the index of a line that joins a pair of clusters. The index can then be used to perform further operations on the line.

The template will ensure that there are lines connecting the clusters. If there are lines missing the shape would not be able to be processed by the template matching stage.

To find the line which connects two clusters we need to find a line which contains POI associated with both of these clusters. This is done by searching through all of the lines. The line is rejected if either of the clusters is not found or if the clusters are separated by a third cluster.

There may be many POI on the same line which are associated with the same cluster. Figure 5.1 shows a line which has POI from two clusters, namely 1 and 2. Our algorithm will return the two points which represent the shortest distance between the two clusters. The section of the line which is limited by these two points is referred to as the 'inner line'.

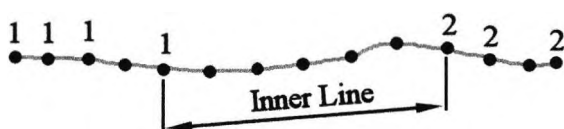


Figure 5.1 – Inner Line

If the specified clusters are separated by a 'false corner' (see Section 4.2.6.1) then extra processing must be performed. Figure 5.2 shows an example of the points which might be present in a drawing. We wish to find the line between cluster 2 and cluster 4. The cluster labelled 'x' has been ignored by the template matching stage. In this case the line that is found will link the start cluster (2) to the false corner cluster (x). Note that in this case the direction of the search is important. The section labelled 'inner line' is the shortest line between the clusters specified.



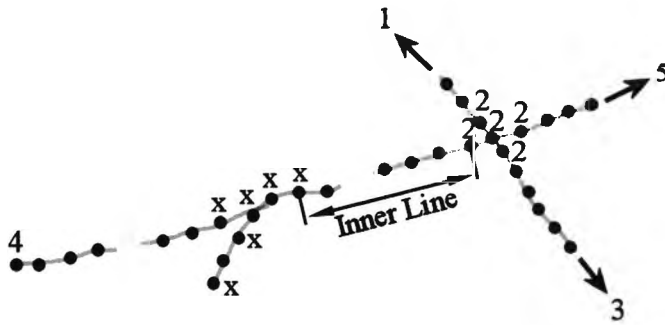


Figure 5.2 – Inner Line with False Corner

This inner line is useful as a starting point to calculating the line lengths.

### 5.3.2 Line Lengths

Once we have a line index we can easily calculate the line length of the entire line. This is performed by simply summing the Euclidean distances between each of the points that make up the line.

Sometimes we need to measure the length of just a section of a line. For example, to calculate the leg lengths of a cross shape. To assess one of the criteria for shape 11 of the VMI test (see Figure 5.3) we need to calculate the length from the intersection of the two lines to each of the line ends.

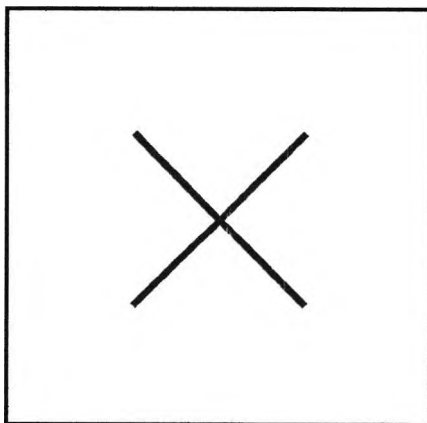


Figure 5.3 - VMI shape 11

In order to calculate leg length we begin by using the calculation described in Section 5.3.1 to find the start and end points of the inner line. The inner line is then extended from the start point until we reach a POI which as the feature type 'Intersection'. The line section is also extended from the end point to the end of the line. The length of this section is then computed by summing the distance between points.

### 5.3.3 Three Cluster Line

For some rules we must find a continuous line which crosses another line. These lines will contain 3 clusters that we are interested in. Figure 5.4 shows a drawing where we are interested in the line which joins clusters 4 and 5.

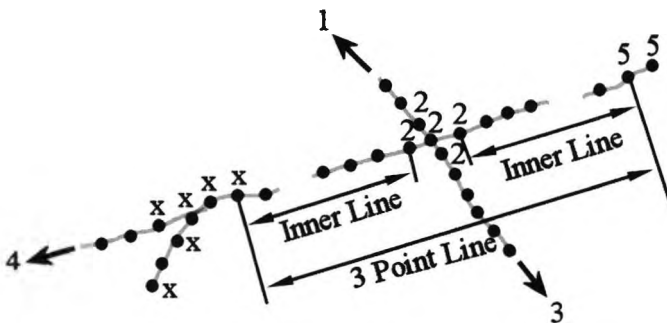


Figure 5.4 - Finding Three Cluster Line

We can consider this line in two halves, specifically:

- A line from cluster 4 to cluster 2.
- A line from cluster 2 to cluster 5.

We find the inner line section of each of these lines using the algorithm described in Section 5.3.1. As we can see in Figure 5.4 this process takes into account false corner clusters.

We can easily ensure that the line is continuous by checking that the line indexes for both inner lines are the same.

We now have four end points which lie on the same line. The points that are at the intersection (cluster 2 in this case) are discarded leaving the end points of the Three Cluster Line.

### 5.4 VMI rules

This section describes each of the rules that were implemented in order to score VMI test drawings. The command syntax is specified for each command. This syntax specifies the command name and any parameters it may take. The syntax, as show here, is also displayed in the user interface when a command is invalid. Figure 5.5 shows a screen shot, of the scoring software, where one of the rules has been incorrectly entered into the template.

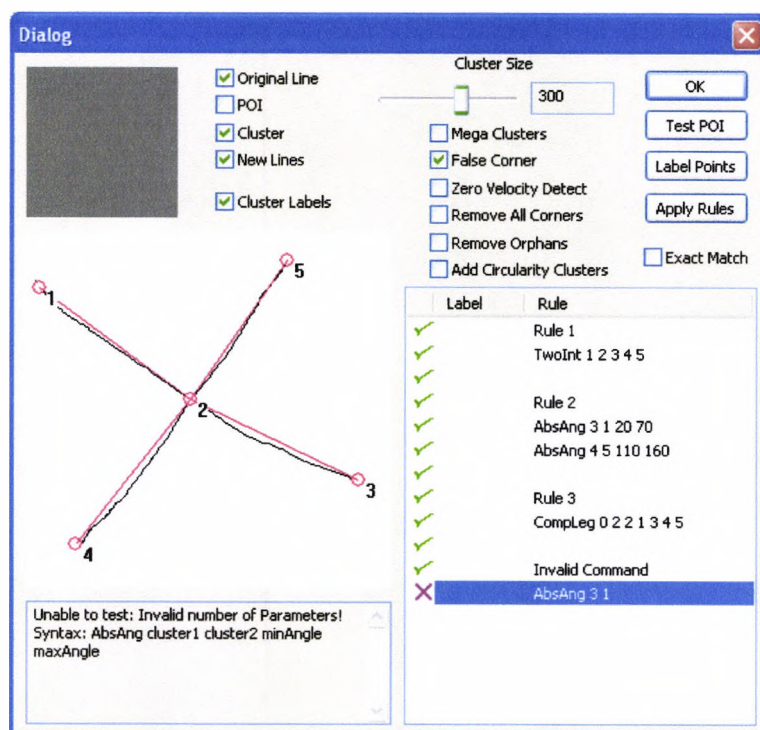


Figure 5.5 - Invalid template showing syntax

In Figure 5.5 the last rule in the list, *AbsAng*, has only two parameters. The correct syntax is displayed below the drawing showing that there must be four parameters specified.

Table 7 shows the set of rules which were used to score the VMI. Each of these rules are described in detail in the following sections.

Rule Name	Scoring Criteria
AbsAng	The angle at which a line is drawn must be within the range specified.
RelAng	The angle between two lines must be within the range specified.
AllLineAng	A given proportion of all of the lines must be within a specified range.
LineAng	A given proportion of a line must be within a specified range.
ClusDist	The proportion of two distances must be within the range specified.
ClusProp	The proportion of longest and shortest, of a set of distances, must be within the range specified.
CompLeg	Compare longest to shortest of a set of legs.
MinLegLength	The length of a leg must be greater than the length specified.
TwoInt	The drawing must consist of two intersecting lines (extensions are permitted)
ThreeInt	The drawing must consist of three intersecting lines (extensions are permitted). The height of the gap, where the lines intersect, must also be below a specified threshold.
Circularity	The points specified must be joined by a circle
SideInt	An imaginary line between two clusters must intersect a third. The outcome of this rule can also be reversed.
CornLine	The over- or under-lap where a corner meets a line must be below a specified threshold.
MustPass	This is a higher rule (see Section 4.4.1.1) which specifies that a minimum number of a set of rules must pass.

**Table 7 - Set of Absolute Rules for VMI**



Each rule can return one of two results, namely:

- “Pass” – the criteria have been verified.
- “Fail” – the criteria has not been met.

In order to be used as a pre-screening tool it is valuable to have a bias towards rejecting drawings. Children who were identified as Dyspraxic by the software would be referred to a clinician for verification. Therefore it is more important to include all borderline cases.

### 5.4.1 AbsAng - Absolute Angles

This rule is used to verify that the angle of a line is within a specified range. Figure 5.6 shows the command syntax for the AbsAng rule.

```
Syntax: AbsAng cluster1 cluster2 minAngle maxAngle
```

**Figure 5.6 – Command syntax for AbsAng**

This rule calculates the slope of a straight line between two clusters specified. This angle is then tested to see whether it is between the min and max angles specified.

For shape 11 of the VMI test to be drawn successfully, one of the lines must be between 110 and 160 degrees. All angles in the VMI manual are specified with zero degrees pointing horizontally to the left.

Figure 5.7 shows an example of a drawing of shape 11. The clusters have been labelled.

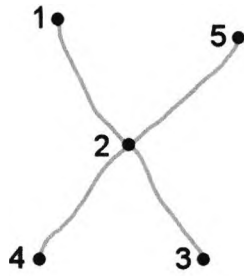


Figure 5.7 - Form 11

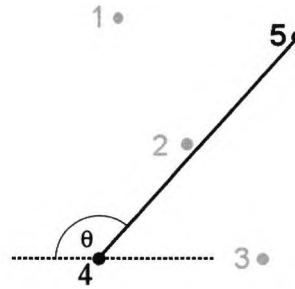


Figure 5.8 - AbsAng Calculation

The line that we need to measure, in this case, will always be between cluster 4 and cluster 5. The command would, therefore, be expressed in the template as:

```
AbsAng 4 5 110 160
```

Where:

- 4 and 5 are the cluster numbers.
- 110 and 160 are the limits of the angle range.

Figure 5.8 illustrates diagrammatically how this rule is assessed. The calculation is performed using only the cluster positions. The angle,  $\theta$ , is calculated from horizontal to an imaginary straight line between the two clusters.

If  $\text{minAngle} \leq \theta \leq \text{maxAngle}$  the rule will return “pass”.

For all other cases it will return “fail”.

The angle calculation and comparison algorithm allows for angle ranges which cross the zero degree line. For example, a range of 330 to 30 degrees is valid.

## 5.4.2 RelAng - Relative Angles

Figure 5.9 shows the command syntax for RelAng.

```
Syntax: RelAng cluster1 cluster2 cluster3 minAngle maxAngle
```

Figure 5.9 – Command Syntax for RelAng

The three clusters specified denote the start, middle and end points of a corner respectively. The angle at the corner (`cluster2`) is calculated. This angle is then tested to see whether it is between the minimum and maximum angles specified.

In shape 20 of the VMI test, one of the scoring criteria is that the left hand outer angle is between 60 and 120 degrees. Figure 5.10 below shows a drawing of this shape after the clusters have been identified.

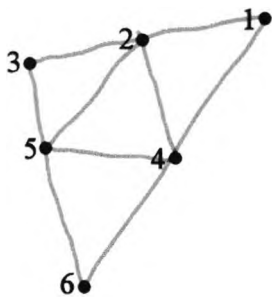


Figure 5.10 - Form 20

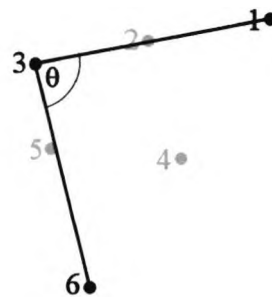


Figure 5.11 – RelAng Calculation

The cluster at the left hand outer angle will always be labelled 3. A command must be created to measure the angle at this point.

We have several choices when selecting the start and end clusters for the command. We can use the nearest clusters, namely 2 and 5, the furthest clusters, namely 1 and 6, or a mixture of one from each.

It is logical to use either both the nearest or both the furthest clusters. In the template for this shape we have used two commands which test both of these possibilities. For the sake of describing the rule, a command using the outer clusters will be analysed. This command would be specified as:

```
RelAng 6 3 1 60 120
```

Where:

- 6 and 1: are the start and end clusters.
- 3: is the corner.
- 60 and 120: are the limits of the angle range.

The calculation is performed using only the cluster positions. Figure 5.11 shows the diagrammatically the angle,  $\theta$ , that we are measuring. This is then compared with the `minAngle` and `maxAngle` specified.

The angle at the corner can be assessed as an acute angle or a reflex angle. The value that is calculated depends on the relative position of the three clusters. We can easily convert between the two angles by calculating  $360 - \theta$ . We cannot assume that the angle range will be acute or reflex, therefore, we compare both angles against the specified range.

If  $\text{minAngle} \leq \theta \leq \text{maxAngle}$  the rule will return “pass”.

If  $\text{minAngle} \leq (360 - \theta) \leq \text{maxAngle}$  the rule will return “pass”.

For all other cases it will return “fail”.

In the template for shape 20 this rule is also used to ensure that the sides of the outer triangle are straight. The commands specified for this are as follows:

```
RelAng 1 2 3 170 190
RelAng 1 4 6 170 190
RelAng 3 5 6 170 190
```

These ensure that the deviation measured at the midpoint is within ten degrees of true.

### 5.4.3 LineAng – Straightness of a line

For some shapes of the VMI test it is specified in the requirements that a certain percentage of the line must be within set angle limits.

For example, one of the scoring criteria for shape7 is:

- At least  $\frac{1}{2}$  of each line must be within 20 degrees of correct angle.

Figure 5.12 shows shape 7 as displayed in the VMI manual.

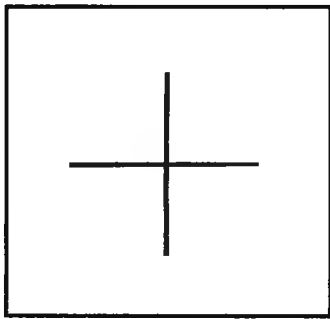


Figure 5.12 – VMI shape 7

Figure 5.13 shows the command syntax for the LineAng rule.

```
Syntax: LineAng cluster1 cluster2 minAngle maxAngle minPercent
```

Figure 5.13 - Command Syntax for LineAng

A line is first identified as described in Section 5.3.1 from the two clusters specified.

Figure 5.14 represents a line made up of 7 points which we will use to describe the algorithm.

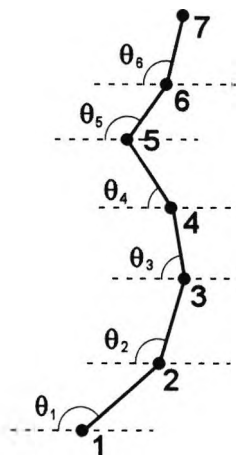


Figure 5.14 – Computing LineAng

For each point (n) in the line the distance to the next point ( $d_n$ ), and the associated angle ( $\theta_n$ ), are calculated. If the angle ( $\theta_n$ ) is within range, the distance ( $d_n$ ) is added to a total ( $T_1$ ). A total ( $T_2$ ) is also kept when angles fall outside the specified range.

If  $\frac{T_1}{T_2} \times 100 \geq \text{minPercent}$  then the rule will return “pass”.

Otherwise the rule will return “fail”.

#### 5.4.4 AllLineAng – Straightness of all lines in a drawing

AllLineAng is a special case of LineAng where all of the lines of the drawing are required to be within a certain range. For the vertical line and horizontal line shapes of the VMI, over half of the lines must be within 30 degrees of the correct angle. Figure 5.15 shows the command syntax for the AllLineAng. We do not need to specify any clusters for this rule as the calculation includes all the lines of the drawing.

```
Syntax: AllLineAng minAngle maxAngle minPercent
```

Figure 5.15 - Command Syntax for AllLineAng

We step through each of the lines. For each one we calculate the length of the line that is within range and the length of the line which is out of range. This is done using the process described in Section 5.4.3.

A total length across all the lines is produced for both the in range and out of range line sections. A percentage is calculated from this and compared against the threshold specified.

### 5.4.5 ClusDist - Proportion of two distances

```
Syntax: ClusDist cluster1 cluster2 cluster3 minProportion
maxProportion
```

Figure 5.16 - Command Syntax for ClusDist

This rule finds the proportion of the length of straight lines from cluster 2 to the other 2 lines. This is then tested against minimum and maximum limits.

For shape 20 of the VMI test, we must verify that the inner triangle touches the outer triangle within the middle third of the line. Figure 5.17 below a drawing of this shape after the clusters have been identified. The points at which the two triangles touch has already been detected as the clusters were created.

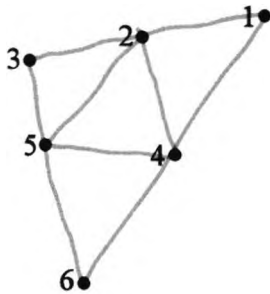


Figure 5.17 - Form 20

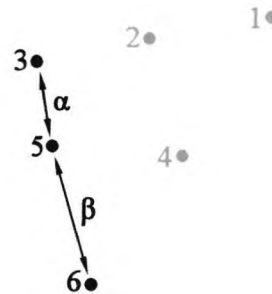


Figure 5.18 - ClusProp Calculation

Considering the left most side only, the command for this would be expressed as:

```
ClusDist 3 5 6 0.5 2
```

Figure 5.18 illustrates the calculation for this command. The lengths  $\alpha$  and  $\beta$  are calculated using the Euclidean distance between the centres of the clusters specified.

If  $0.5 \leq \frac{\alpha}{\beta} \leq 2$  the rule will return “pass”.

Otherwise the rule will return “fail”.

Because we only use the location of the clusters to perform the calculation, this rule can be used to compare distances even if there is no line between the clusters. This is useful in shape 17, see Figure 5.19, where we need to check that there is at least 2 to 1 spacing between circles. This is illustrated in Figure 5.20.

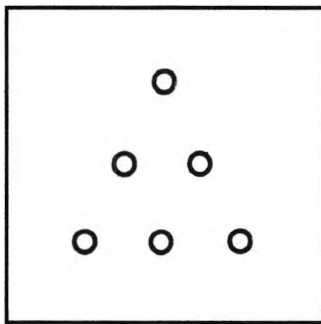


Figure 5.19 - VMI shape 17

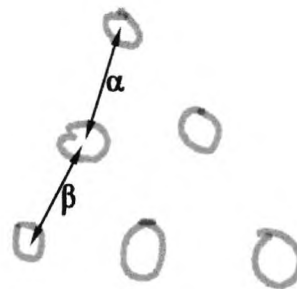


Figure 5.20 - ClusDist applied to shape 17

#### 5.4.6 ClusProp – Proportion of longest to shortest distance

This rule can be used to compare distances on the drawing. The distances are measured between clusters, therefore, two clusters must be specified for each distance. For each cluster pair the (Euclidean) distance between them is calculated. The proportion between the longest and shortest distance is then calculated. A minimum and maximum threshold can be specified to compare it to. Figure 5.21 shows the syntax for this rule. This differs from the ClusDist rule in that more than two distances can be measured and they do not need to share a common cluster.

```
Syntax: ClusProp atLeast noMore cluster1 cluster2 ...
```

Cluster pairs identify lines to be compared (as many as needed can be added).

Proportion between longest and shortest line (max/min) must be at least 'atLeast' and no more than 'noMore'!

(A 'Zero' value for atLeast or noMore indicates that that threshold will be ignored)

Figure 5.21 - Command Syntax for ClusProp



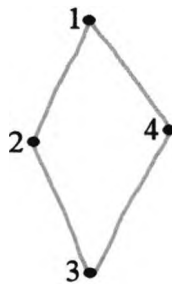


Figure 5.22 – VMI Shape 19.

Figure 5.22 shows a drawing copied from shape 19 of the VMI test. One of the criteria for this shape is that the shortest side is at least  $\frac{2}{3}$  of the longest side. In order to create a command to assess this criterion we must first specify the sides. These can be specified in any order but we have started from the top and worked anti-clockwise. Four pairs of clusters are specified, namely:

- Distance 1 – Cluster 1 to Cluster 2.
- Distance 2 – Cluster 2 to Cluster 3.
- Distance 3 – Cluster 3 to Cluster 4.
- Distance 4 – Cluster 4 to Cluster 1.

The VMI criterion specifies a ratio of a least  $\frac{2}{3}$ . We could set *atLeast* to 0.6666, but a more accurate result is calculated by setting *noMore* to 1.5 ( $\frac{3}{2}$ ). By setting *atLeast* to zero the test will only be performed on *noMore*.

The command specified for the criteria specified would be:

```
ClusProp 0 1.5 1 2 2 3 3 4 4 1
```

To clarify this Figure 5.23 shows the command broken down into component parts.

Rule Name	atLeast	NoMore	Distance 1	Distance 2	Distance 3	Distance 4
Clus Prop	0	1.5	1 2	2 3	3 4	4 1

Figure 5.23 - Break down of ClusPop command

Figure 5.24 shows the distances which will be measured.

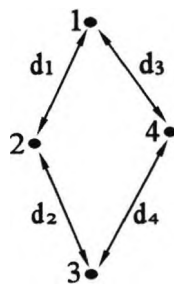


Figure 5.24 - ClusProp Calculation

If  $\frac{\max(d_1, d_2, d_3, d_4)}{\min(d_1, d_2, d_3, d_4)} \leq 1.5$  the rule will return “pass”.

Otherwise it will return “fail”.

### 5.4.7 CompLeg - Compare longest to shortest leg

```
Syntax: CompLeg atLeast noMore centreCluster tipCluster1 ...
```

Proportion between longest and shortest leg (max/min) must be at least 'atLeast' and no more than 'noMore'!

(A 'Zero' value for atLeast or noMore indicates that that threshold will be ignored)

Figure 5.25 - Command Syntax for CompLeg

CompLeg is a rule which compares the lengths of a set of lines. In this rule all the lines must join to a central cluster. The centre cluster is specified followed by a list of the connected clusters. The command syntax for this rule is shown in Figure 5.25.

The thresholds for CompLeg are tested in a similar fashion as for ClusProp (see Section 5.4.6). This rule, however, calculates the length of a line from the sample points rather than a distance between clusters. The algorithm for calculating the leg length is described in Section 5.3.2.

An example of a use of this rule is found in shape 11 of the VMI. One of the criteria states that the longest leg should be no more than twice the length of the shortest leg.

Figure 5.26 shows a copied version of the shape with the clusters labelled. Figure 5.27 shows a different example of same shape. In Figure 5.27 one of the legs in the drawing has been extended with a second line. The cluster marked with an x has been removed as a 'false corner' (see Section 4.2.6.1).

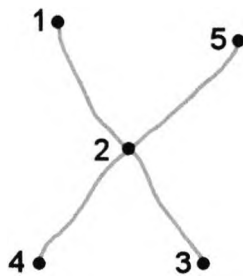


Figure 5.26 – VMI shape 11

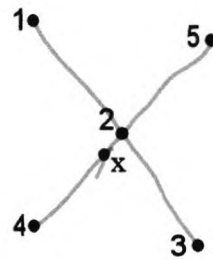


Figure 5.27 - Form 11 with extension

The extended leg in Figure 5.27 will be taken into account by the leg length algorithm.

The CompLeg command which would be used to assess shape 11 of the VMI could be expressed as:

```
CompLeg 0 2 2 1 3 4 5
```

Once all of the leg lengths have been calculated the proportion of maximum and minimum is calculated and compared against the thresholds stated.

### 5.4.8 MinLegLength – Minimum Leg Length

```
Syntax: MinLegLength minLength centreCluster tipCluster
```

**Figure 5.28 - Command Syntax for MinLegLength**

A leg is specified by a centre cluster and a tip cluster. The length is calculated using the leg length algorithm described in Section 5.3.2.

This distance is compared against the `MinLegLength` value specified. If it is smaller the rule returns “fail” otherwise it returns “pass”.

### 5.4.9 TwoInt - Two intersecting lines

```
Syntax: TwoInt cluster1 cluster2 cluster3 cluster4 cluster5  
c1->c2->c3 is one line  
c4->c2->c5 is the second line
```

**Figure 5.29 - Command Syntax for TwoInt**

`TwoInt` is used to check that the shape is made up of two intersecting lines. Figure 5.29 shows the command syntax for this rule. Figure 5.30 shows a drawing to which this rule might be applied, this is a copy of the VMI shape 11. One of the criteria for shape 11 is that it is made from two intersecting lines. Extensions to the line are allowed.

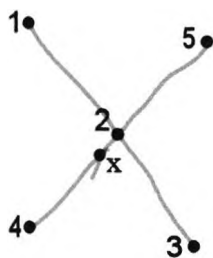


Figure 5.30 - Shape 11 with extension

The command for this example would be:

```
TwoInt 1 2 3 4 5
```

First we find each of the lines which go through the centre cluster (cluster 2 in this example). Because extensions to the lines are allowed, we use the Three Cluster Line algorithm, described Section 5.3.3, for this. The rule returns “fail” if it finds more than two lines.

We have now found two line sections which connect the clusters specified. Both of these line sections are tested to ensure that they only contain one intersection. The rule returns “pass” if this is the case.

#### 5.4.10 ThreeInt - Three intersecting lines

ThreeInt ensures that a drawing is made up of three intersecting lines. This is similar to TwoInt (see Section 5.4.9). Three lines are unlikely to cross at exactly the same point, therefore, a triangular gap is formed at the centre of the shape. The height of this is calculated. Figure 5.31 shows a drawing of VMI shape 14, the crossing point is magnified to show the gap. For shape 14 of the VMI this height must be less than 1/8 of an inch in order to meet the test criteria.

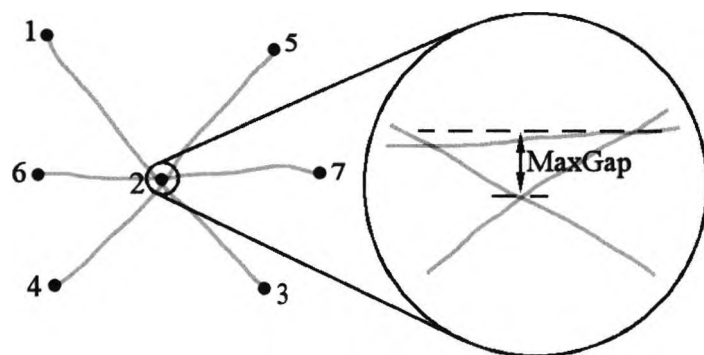


Figure 5.31 - Form 14 showing intersection gap

Figure 5.32 shows the command syntax for ThreeInt. The centre cluster must be shared by all three lines.

```
Syntax: ThreeInt cluster1 cluster2 cluster3 cluster4 cluster5
cluster6 cluster7 maxGap

c1->c2->c3 is one line
c4->c2->c5 is another line
c6->c2->c7 is the last line
maxGap is the maximum vertical opening allowed
```

Figure 5.32 - Command Syntax for ThreeInt

The Three Cluster Line algorithm described in Section 5.3.3 is used to find the three lines, specifically:

- Line 1 – Cluster 1 to Cluster 3.
- Line 2 – Cluster 4 to Cluster 5.
- Line 3 – Cluster 6 to Cluster 7.

Then they are each tested to make sure that they only intersect once with each of the other two lines. They also must not intersect with any other lines.

The points of intersection are recorded and the greatest difference in vertical position is calculated. The rule returns “pass” if this difference is smaller than *maxGap* and the number of intersections is correct.

#### 5.4.11 Circularity - The points specified must be joined by a circle

This rule is used to assess the circularity of a set of lines. The lines may be connected or may have gaps between them. In shape 24 of the VMI (see Figure 5.33) breaks in the circular lines are required.

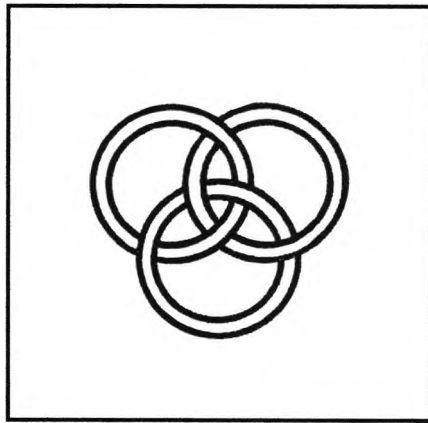
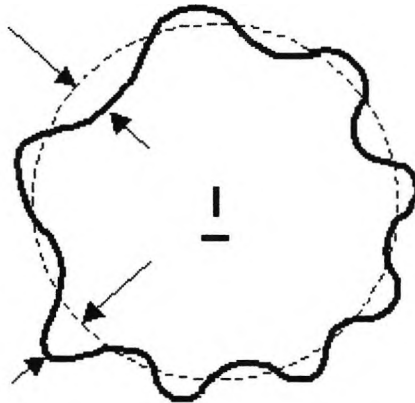


Figure 5.33 - VMI Shape 24

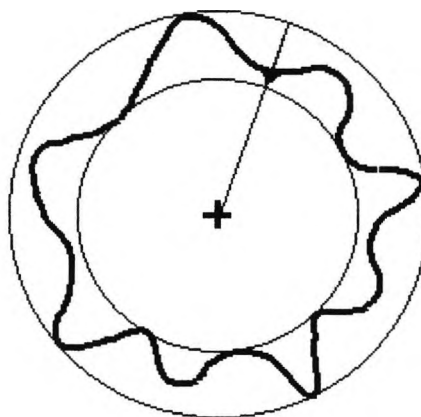
Several algorithms to detect circles using image processing techniques are described in the literature. Ioannuo, Huda and Laine (1999) describe a two step algorithm which uses a 2D Hough Transform to find the circle centres. Chiu and Liaw (2004) show how the computation and storage requirements of the Hough Transform can be reduced using a voting method. Chen and Chung (2001) describe an algorithm which avoids using a Hough transform by use of a Randomized algorithm. Qiao and Ong (2004) have developed an algorithm which utilises the connections between points to detect multiple circles.

Given that we have a geometric description of our lines, several standard measures of circularity can be applied to the shape. Four such measures are described in the literature.



**Figure 5.34 - Least Squares Reference Circle (LSC)**

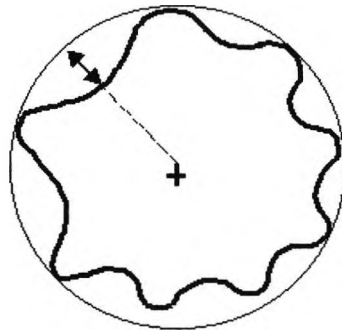
Figure 5.34 depicts the Least Square Reference Circle (LSC). This is a circle where the sum of areas inside this circle are equal to the sum of the areas outside the circle and kept to a minimum separation.



**Figure 5.35 - Minimum Zone Circle (MVC)**

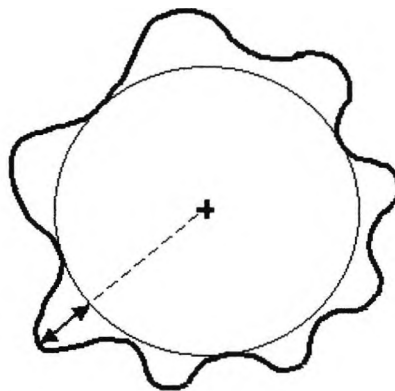
Figure 5.35 depicts the Minimum Zone Circle (MVC). This is defined as two concentric circles positioned to just enclose the measured profile such that their radial departure is a minimum. The roundness value is then given as their radial separation.





**Figure 5.36 - Minimum Circumscribed Circle (MCC)**

Figure 5.36 depicts the Minimum Circumscribed Circle (MCC). This is also known as the ring gauge reference circle and is the smallest circle that totally encloses the profile. Out of roundness is quantified as the largest deviation from this circle



**Figure 5.37 - Maximum Inscribed Circle (MIC)**

Figure 5.37 depicts the Maximum Inscribed Circle (MIC). This is sometimes referred to as the plug gauge circle and is the largest circle that is totally enclosed by the profile. Errors are quantified as the maximum radial deviation away from this reference circle.

Zhu, Ding and Xiong (2003) have used a steepest descent algorithm to compute all of these geometric measurements. Novaski and Barczak (1997) employed Voronoi diagrams to compute LSC and MZC. A robust algorithm to compute the LSC is described by Kim and Kim (1995). Wang et al. (1999) have used an infinitesimal

analysis method to compute the MZC. Swanson, Lee and Wu (1995) developed an optimal time algorithm to compute MZC. Jywe et al. (1999) have assessed the MCC, MIC and MZC using simultaneous linear algebraic equations. Dhanish (2002) has developed a simple algorithm to calculate MZC. Samuel and Shunmugam (2000) have used computational geometric concepts to determine the MZC. Pilu, Fitzgibbon and Fisher (1997) describe a Least Squares algorithm which matches an ellipse to the data rather than a circle.

We have implemented the Minimum Circumscribed Circle and Maximum Inscribed Circle measurements. The command syntax is shown in Figure 5.38.

```
Syntax: Circularity minResult maxResult cluster1 ... clusterN
('-' used for gaps in circle)
2 clusters minimum required
```

Figure 5.38 - Command syntax for Circularity

Figure 5.39 and Figure 5.40 show this rule applied to VMI shape 13.

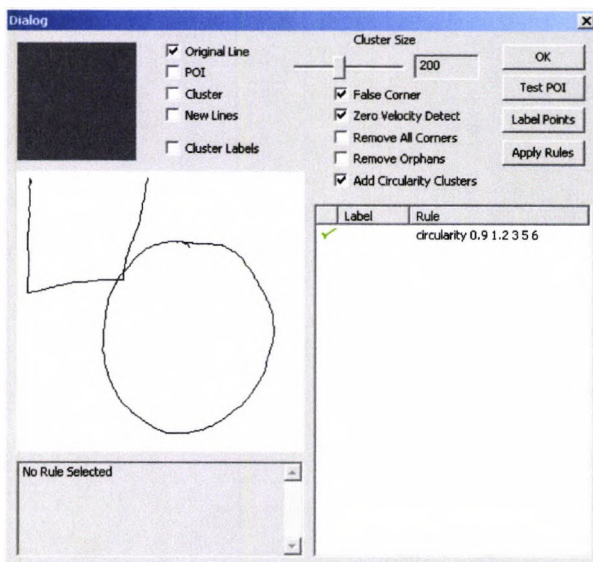


Figure 5.39 - VMI Shape 13

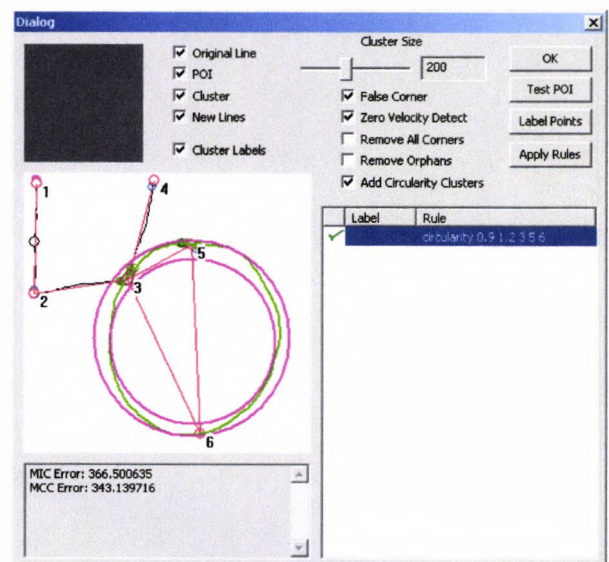


Figure 5.40 - MIC and MCC circles

### 5.4.12 SideInt - Line between two clusters intersects a third

SideInt calculates whether a straight line drawn between two clusters would intersect with any lines containing points from a third cluster. Figure 5.41 shows the command syntax for this rule.

```
Syntax: SideInt touch cluster1 cluster2 cluster3

touch specifies whether the line c1->c2 should
hit (touch!=0) or miss (touch==0) cluster3.
```

Figure 5.41 - Command Syntax for SideInt

The command specifies whether the line should or should not intersect with the middle cluster.

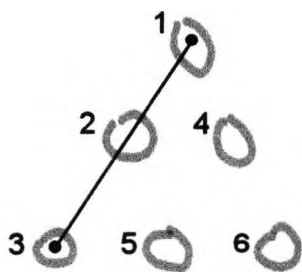


Figure 5.42 - Form 17 showing SideInt

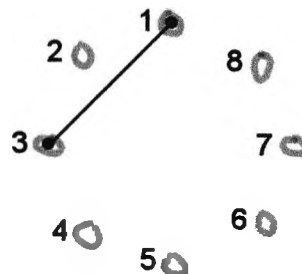


Figure 5.43 - Form 21 showing SideInt

Figure 5.42 shows the rule being used in shape 17 of the VMI. To ensure that the side is straight we test to see that a line drawn between the centres of clusters 1 and 3 passes through any point of cluster 2. The command for this would be:

```
SideInt 1 1 3 2
```

Figure 5.43 shows the rule being used in shape 21 of the VMI. The scoring criteria states that no three centres lie on a straight line. The SideInt rule is used to ensure that

a straight line between the centres of clusters 1 and 2 does not intersect with cluster 3. The command for this would be:

```
SideInt 0 1 3 2
```

### 5.4.13 CornLine - Max error when corner touches a line

Figure 5.44 shows a drawing of VMI shape 20. This shape can be considered as two triangles, one inside the other. Where the two triangles meet there may be an overlap (see Figure 5.45) or an underlap (see Figure 5.46). We do not, however, know whether there will be an overlap or an underlap beforehand.

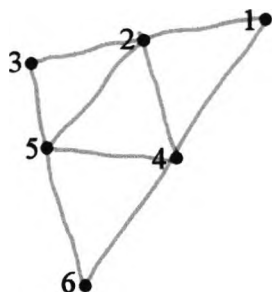


Figure 5.44 - Form 20

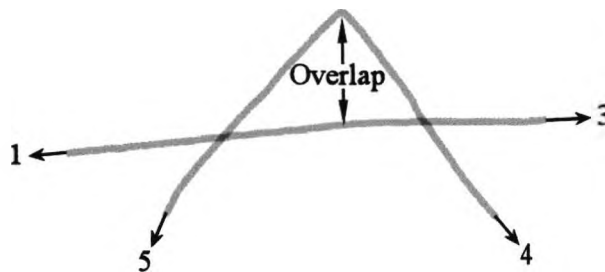


Figure 5.45 - Close up of Cluster 2 showing overlap

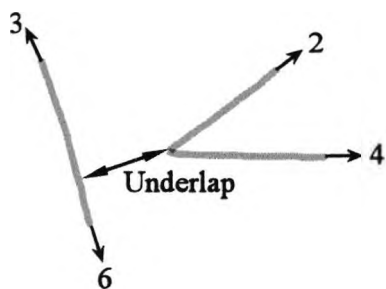


Figure 5.46 - Close up of Cluster 5 showing underlap

CornLine finds the error where a corner meets a line. The over or underlap is calculated and this is tested against a threshold specified.

```
Syntax: CornLine cluster1 cluster2 cluster3 cluster4 cluster5 maxGap
Clusters 1->2->3 are the line
Clusters 4->2->5 are the corner
```

Figure 5.47 - Command Syntax for CornLine

The first step is to find the two lines to test. These lines may be continuous or may be made up of two lines. For single lines the Three Cluster Line algorithm described in Section 5.3.3 is used. Otherwise the line sections are concatenated to produce a continuous line.

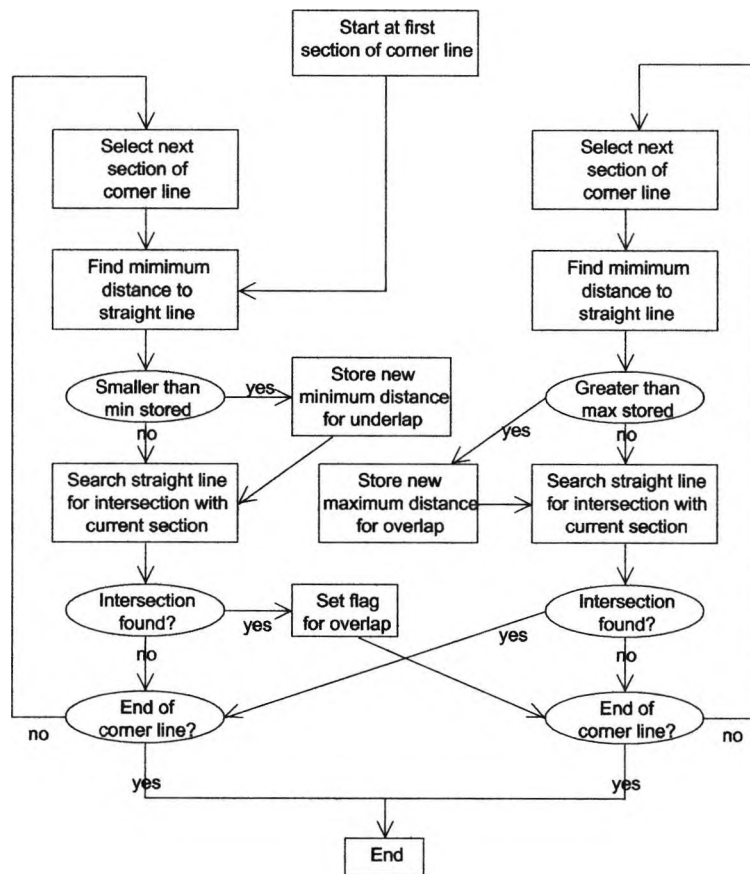


Figure 5.48 - Flow diagram of CornLine algorithm

Figure 5.48 shows a flow diagram for the algorithm. We know there is an overlap if there are intersections between the two lines. This is ascertained by scanning through the corner line. At each point of the corner line we scan through the straight line looking for the closest point and also to check for intersections. For underlaps we are

looking for the lowest value for the closest point. For overlaps we are looking for the highest value for the closest point.

If the overlap flag is set then the result is the max distance value. If the overlap flag is not set then the result is the min distance value.

This rule is used several times in scoring shape 20. We check that all the corners of the inner triangle are within 1/3" of the outer triangle. Then we check all three at 1/16". The second set of tests is arranged using MustPass so that only 2 of the corners need to be within this threshold.

#### **5.4.14 MustPass – The number of rules that must pass**

MustPass is a higher level rule. See Section 4.4.1.1 for a description of higher rules. A set of rules is specified by listing the rule labels. The number of rules that must return "pass" is also specified. The command syntax for this rule is shown in Figure 5.49.

```
Syntax: MustPass minPass ruleLabel1 ruleLabel2 ...
```

```
minPass is the number of rules that must pass.  
This is followed by a list of rule labels.
```

**Figure 5.49 - Command Syntax for MustPass**

## 5.5 Templates

A template must be produced for each of the shapes in VMI test.

We now describe the template which has been defined for shape 20 of the VMI test, the Tilted Triangles (see Figure 5.50).

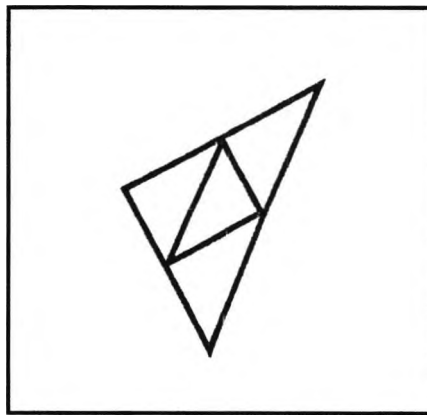


Figure 5.50 - VMI shape 20

### 5.5.1 General Information

The first section of the template provides information about the template. Figure 5.51 shows the layout of this section. We have given the template a name based on the shape that it is designed to assess. Also the name of the author has been stored here.

```
// -- General Information --  
  
Template Name : (20) Tilted Triangles v1.0  
Created By : Timothy Boyle 20/12/2004  
  
// -- End General Information --
```

Figure 5.51 - General Information

The software will ignore any other lines of text which are present in this section. It is possible, therefore, to enter extra information. A version history could be stored, for example, listing changes made to the template.

### 5.5.2 VMI Criteria

The VMI Criteria section does not affect the assessment of the drawings. It is included to describe the set of conditions that the template has been designed to verify. Figure 5.52 shows the layout of this section. We have specified the number of criteria. A textual description for each criterion then follows. The end of each criterion is marked by the end of line character.

```
// -- VMI Criteria --  
  
Number of Criteria : 4  
Two Triangles  
2 corners cleanly touch middle 1/3 of outer triangle (3rd within  
1/16").  
Left outer angle within 60-120 degrees  
Right outer side slopes 100 degrees or more  
  
// -- End VMI Criteria --
```

**Figure 5.52 - VMI Criteria**

The scoring component parses the criteria and stores each one in a separate text string. Functionality to access these strings is provided. The current test harness does not make use of this information, however, if the component was included into a different program this information could be used, for example, to display as part of a user interface.

### 5.5.3 POI Identifier

This section of the template determines the options which are used whilst calculating the positions of clusters in a drawing.

```
// -- POI Identifier --  
  
Cluster Size : 200  
False Corner : 1  
Zero Velocity Detect : 1  
Remove All Corners : 0  
Remove Orphans : 0  
Add Circularity Clusters : 0  
Mega Clusters : 0  
  
// -- End POI Identifier --
```

**Figure 5.53 - POI Identifier**



Figure 5.53 shows the layout of this section. These parameters are described in Section 4.2.

Any line which is not recognised will be ignored by the software. Also, the software will set a default value for any option which not specified in the file. If later versions of the software add extra parameters, we will still be able to use the same files.

### 5.5.4 Template Matching

This section of the template defines the connections between the expected clusters. Clusters cannot exist, in the ideal shape, without connecting to another cluster. This means that it is not necessary to specify the number of expected clusters as this can be deduced from the connections. Figure 5.54 shows the numbers which are assigned to the features of the shape. Figure 5.55 shows the template matching section.

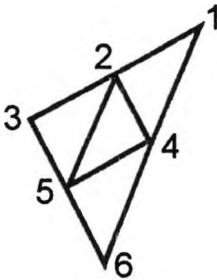


Figure 5.54 - Shape 20 Template

```
// -- Template Matching --
Number of rules 11
1c2
1c4
2c4
2c3
2c5
3c5
4c5
4c6
5c6
1a3
3a6

// -- End Template Matching --
```

Figure 5.55 - Template Matching

### 5.5.5 Absolute rule section

The absolute rule section lists the commands which will be used to verify that the drawing meets the criteria specified in the VMI test manual. The format of this section of the template is described in detail in the previous chapter (see Section 4.4).

Figure 5.56 shows the commands which we have used to verify the VMI criteria for shape 20.

```
// -- Absolute Rule Section --

// Form 10: Two Triangles

// 1) Two Triangles
Comment Rule 1)
RelAng 1 2 3 170 190
RelAng 1 4 6 170 190
RelAng 3 5 6 170 190
Comment

// 2) 2 Touch 1/3 (3rd 1/16")
Comment Rule 2)
ClusDist 1 2 3 0.5 2
ClusDist 1 4 6 0.5 2
ClusDist 3 5 6 0.5 2
// All inner corners must touch within 1/16"
CornLine 1 2 3 4 5 158
CornLine 1 4 6 2 5 158
CornLine 3 5 6 2 4 158
// Two inner corners must touch 'exactly'
labelCorner2 CornLine 1 2 3 4 5 80
labelCorner4 CornLine 1 4 6 2 5 80
labelCorner5 CornLine 3 5 6 2 4 80
MustPass 2 Corner2 Corner4 Corner5
Comment

// 3) 60 - 120 degrees Left
Comment Rule 3)
RelAng 2 3 5 60 120
RelAng 1 3 6 60 120
Comment

// 4) 100+ degrees slope
Comment Rule 4)
AbsAng 4 1 100 180
AbsAng 6 4 100 180
AbsAng 6 1 100 180

// -- End Absolute Rule Section --
```

Figure 5.56 - Absolute Rule Section

## 5.6 Results

In order to test the performance of the software we first needed a set of scores to compare against. These were generated manually using the VMI Scoring Assistant software described in Chapter 3. It is important to note that the manual scoring of these drawings was performed by the author rather than by a trained clinician. See Section 2.5 for details.

Scoring was then performed using the General Scoring software as described in this chapter. For each drawing the appropriate template was used to produce a pass or fail decision. The test harness (see Section 4.1.4) allows the user to select all drawings of a particular shape from the database. A template can then be chosen for this shape. The tests can then be run with one mouse click. Comparing the results of the test run with the results stored in the database is also performed by the software. The results of this comparison are summarised in a dialog box.

The results produced by the software do not always agree with the results from manually assessing the VMI criteria. We have categorised these mismatches into two groups, specifically:

- False negative: The drawing meets all of the criteria specified in the VMI manual, however, the software could not verify the shape against the template.
- False positive: The drawing is successfully verified by the software, but fails one or more of the VMI scoring criteria.

Test	Shape	Samples	False Positive	False Negative	Correct (%)
4	Vertical Line	45	0	0	100
5	Horizontal Line	45	0	0	100
6	Circle	45	0	2	95.6
7	Vertical-Horizontal Cross	46	0	1	97.8
8	Right Oblique Line	46	1	0	97.8
9	Square	46	1	3	91.3
10	Left Oblique Line	45	2	0	95.6
11	Oblique Cross	45	0	0	100
12	Triangle	45	0	5	88.9
13	Open Square and Circle	46	8	5	71.7
14	Three-Line Cross	46	0	2	95.7
15	Directional Arrows	45	1	16	62.2
16	Two Dimensional Rings	45	1	13	68.9
17	Six Circles	45	3	4	84.4
18	Circle and tilted Square	43	1	9	76.7
19	Vertical Diamond	43	1	7	81.4
20	Tilted Triangles	43	0	7	83.7
21	Eight-Dot Circle	35	4	1	85.7
22	Wertheimer's Hexagons	34	0	10	70.6
23	Horizontal Diamond	35	1	5	82.9
24	Three-Dimensional Rings	29	0	6	79.3
25	Necker Cube	30	2	2	86.7
26	Tapered Box	29	2	6	72.4
27	Three-Dimensional Star	29	0	1	96.6

Table 8 - Control Subjects Success Rate

Table 8 shows the results for control subjects. Shapes 1, 2 and 3 of the VMI test were not performed by the control subjects as they were all able to draw these shapes without assistance.

Test	Shape	Samples	False Positive	False Negative	Correct (%)
1	Vertical Line	3	0	0	100
2	Horizontal Line	3	0	0	100
3	Circle	3	0	1	66.7
4	Vertical Line	31	0	0	100
5	Horizontal Line	31	0	0	100
6	Circle	31	0	5	83.9
7	Vertical-Horizontal Cross	31	1	0	96.8
8	Right Oblique Line	31	0	0	100
9	Square	31	0	8	74.2
10	Left Oblique Line	31	0	1	96.8
11	Oblique Cross	31	0	0	100
12	Triangle	31	1	1	93.5
13	Open Square and Circle	31	3	4	77.4
14	Three-Line Cross	31	0	1	96.8
15	Directional Arrows	27	2	3	81.5
16	Two Dimensional Rings	27	0	8	70.4
17	Six Circles	27	4	0	85.2
18	Circle and tilted Square	19	1	3	78.9
19	Vertical Diamond	19	0	0	100
20	Tilted Triangles	19	1	1	89.5
21	Eight-Dot Circle	15	1	1	86.7
22	Wertheimer's Hexagons	15	0	0	100
23	Horizontal Diamond	15	1	0	93.3
24	Three-Dimensional Rings	11	0	3	72.7
25	Necker Cube	11	0	2	81.8
26	Tapered Box	11	3	1	63.6
27	Three-Dimensional Star	11	0	1	90.9

Table 9 - Known Dyspraxic Success Rates

Table 9 shows the results for known Dyspraxic subjects.

Test	Shape	Correct (%)	False Positive (%)	False Negative(%)
1	(Copied) Vertical Line	100		
2	(Copied) Horizontal Line	100		
3	(Copied) Circle	66.6		33.3
4	Vertical Line	100		
5	Horizontal Line	100		
6	Circle	90.7		9.2
7	Vertical-Horizontal Cross	97.4	1.3	1.3
8	Right Oblique Line	98.7	1.3	
9	Square	84.4	1.3	14.3
10	Left Oblique Line	96.1	2.6	1.3
11	Oblique Cross	100		
12	Triangle	90.8	1.3	7.9
13	Open Square and Circle	70.1	13	16.9
14	Three-Line Cross	96.1		3.9
15	Directional Arrows	72.2	4.2	23.6
16	Two Dimensional Rings	69	1.4	29.6
17	Six Circles	84.4	9.4	6.3
18	Circle and tilted Square	77.4	3.2	19.4
19	Vertical Diamond	87.1	1.6	11.3
20	Tilted Triangles	85.4	1.6	12.9
21	Eight-Dot Circle	87	6.5	6.5
22	Wertheimer's Hexagons	75		24.4
23	Horizontal Diamond	86	4	10
24	Three-Dimensional Rings	77.8		22.2
25	Necker Cube	85.4	4.9	9.8
26	Tapered Box	70	12.5	17.5
27	Three-Dimensional Star	97.2		2.8

Figure 5.57 – Overall Success Rates

Figure 5.57 shows the overall scoring accuracy that was attained for each shape of the VMI test.

## **5.7 Conclusions**

We have extended the general scoring software and created templates which allow us to assess all of the shapes from the Beery Developmental Test of Visual-Motor Integration.

The scoring decisions made have been compared to those made by a human. The results of this initial study indicate that development of a fully automated system is obtainable.

The scoring produced by the software does not always agree with a manual assessment of the shape. Due to the highly objective criteria that it uses, false negatives are more likely than false positives. If this software were to be used as part of a screening process this would be desirable behaviour. The purpose of such a software system would be to narrow down a set of children, rather than perform a diagnosis.

# Chapter 6

## Dynamic Movement Analysis

### **6.1 Introduction**

Using on-line data capture allows us the opportunity to analyse children's drawings in ways which would not be possible from a pen on paper drawing. In this chapter we investigate new methods of analysis which may lead to a greater understanding of the nature of Dyspraxia.

Rodger et al. (2003) noted the importance of clinician's observations made during the conduct of standardized assessments such as the VMI. Strategy use and time taken to complete the test can be useful indicators of movement and performance issues.

Some studies have argued that kinaesthetic dysfunction is a major component of the problem in Dyspraxia. Mon-Williams et al. (1999) note that as many tests (such as the VMI) are based around a series of static judgements, they do not measure kinaestheses. Dynamic analysis addresses this issue by allowing kinaesthetic measurements to be taken.

Several studies have suggested that Dyspraxic children will perform the copying task in a shorter time. Dewey (1995) says that Dyspraxic children prefer a more ballistic movement strategy that is less dependant upon visual correction. The disorder is



characterised by faster and cruder movements, lack of inhibition of co-movements and poor co-ordination of fine motor skills. Smits-Engelsman et al. (2001) suggest that enhanced phasic stiffness of the limb system results in higher movement velocity and fewer velocity peaks.

Conversely, Rodger et al. (2003) found the time taken to complete the VMI was longer for children with Dyspraxia than for their classmates. Maruff et al (1999) state that the slowing of motor performance is characteristic, and, indeed, diagnostic of the disorder.

Faster drawing could also be due to an attention deficit comorbidity which causes the child to rush the drawing.

There has been a lot of work done in the field of modelling hand movements. It is hoped that by applying these techniques to both patient and control drawings we will be able to discriminate between patient and control subjects. Also it is hoped that this will shed some more light as to the underlying nature of the disorder.

### **6.2 Movement Modelling**

Movement modelling allows a drawing to be reconstructed by applying a set of parameters to a mathematical model of the drawing system. By matching a reconstruction to an existing drawing we can find a set of parameters that describes the original drawing. These parameters may relate directly to attributes of the physical system which is modelled or may just be an abstract feature of the model. By modelling the VMI drawings we may find parameters which relate to the muscular system and some which relate to the motor commands issued by the higher brain functions.

Plamondon (1991) set out three criteria that must be met by such a modelling system.

1. It should rely on a small number of significant parameters to describe pen tip movements.

2. These parameters should allow reconstruction of tip movement within the limits of experimental error.
3. The overall parameter extraction should run automatically.

Hollerbach (1981) produced a model based on the theory that all the basic shapes present in cursive handwriting can be produced by modulating the relative phase, frequency and amplitude of two orthogonal oscillators. This however has only been used to model simple movements and has not reached a stage where a whole drawing or piece of handwriting can be modelled. Lelivelt et al. (1988) has shown how this model relates to the physiology of pen movements, by mapping the vertical and horizontal axis to movements of the fingers and wrist respectively.

Plamondon (1995a, 1995b, 1997) describes a model known as the delta log-normal law in a set of three papers. The model is based on Rapid Aimed Movements (RAM). This class of movements is characterised by a single bell-shaped, asymmetric velocity profile. Plamondon's model produces very respectable empirical results when reproducing Rapid-Aimed movements.

The method we have chosen to model our VMI drawing is the theory of rapid aimed movements.

### **6.3 Rapid Aimed Movements**

A complex movement can be assumed to be made from a series of smaller simple movements. The simplest movement is known as a Rapid-Aimed Movement. By plotting the velocity of the pen over time we get a graph which is approximately bell shaped. By reproducing this curve mathematically we can extract a set of modelling parameters. It is potentially useful to determine whether certain parameters will differ consistently between Dyspraxic patients and Control subjects, thus providing us with new insights into the condition of Dyspraxia, and offering the possibility of developing new tools to evaluate and analyse the condition.

Plamondon (1995a) has developed a mathematical model for a simple rapid aimed movement. A ballistic movement can be simulated mathematically using Equation 1.

$$v(t) = \frac{\bar{v}(t_1 - t_0)}{\sigma\sqrt{2\pi}(t - t_0)} \exp - \left\{ \left[ \ln \left( \frac{t - t_0}{t_1 - t_0} \right) - \mu \right]^2 \cdot \frac{1}{2\sigma^2} \right\}$$

Equation 1 - Delta Log-normal

Equation 1 has been implemented in a Microsoft Excel spreadsheet and the calculation verified against Plamondon's results. Figure 6.1 shows a velocity curve produced by the spreadsheet.

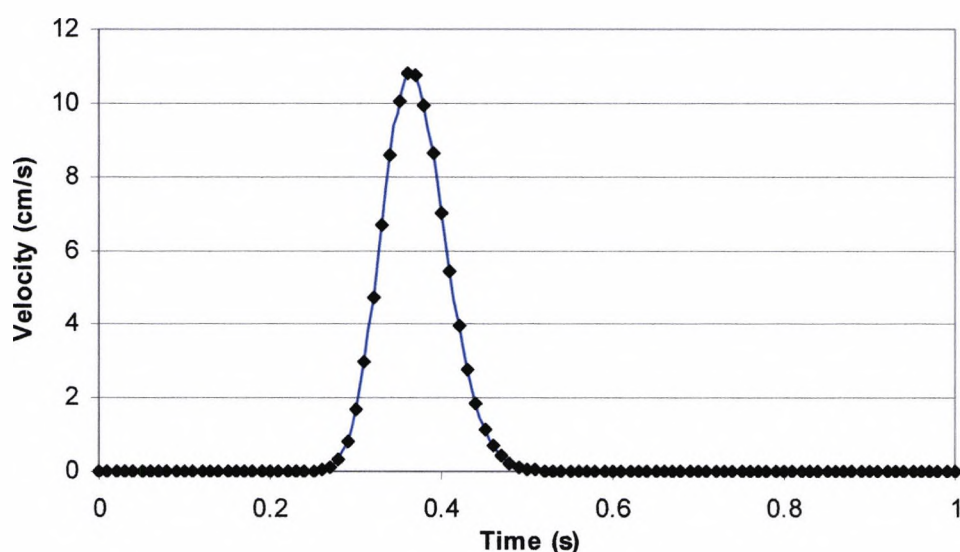


Figure 6.1 - Modelled Velocity Curve.

By finding the correct parameters it is possible to match a modelled velocity curve with data collected from actual human movements.

Plamondon (1995a) goes on to show that even the simplest movements require the coordination of several skeletal muscles acting in groups rather than individually. Such a group is generally referred to as a synergy. The muscles that cause the desired action are called agonists and those causing the opposite effect, antagonists. Figure 6.2 shows a schematic view of the neuromuscular synergy involved in the production of a rapid-aimed movement.

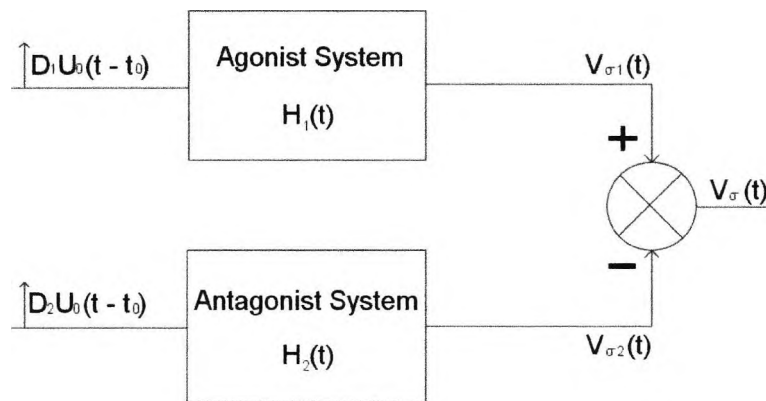
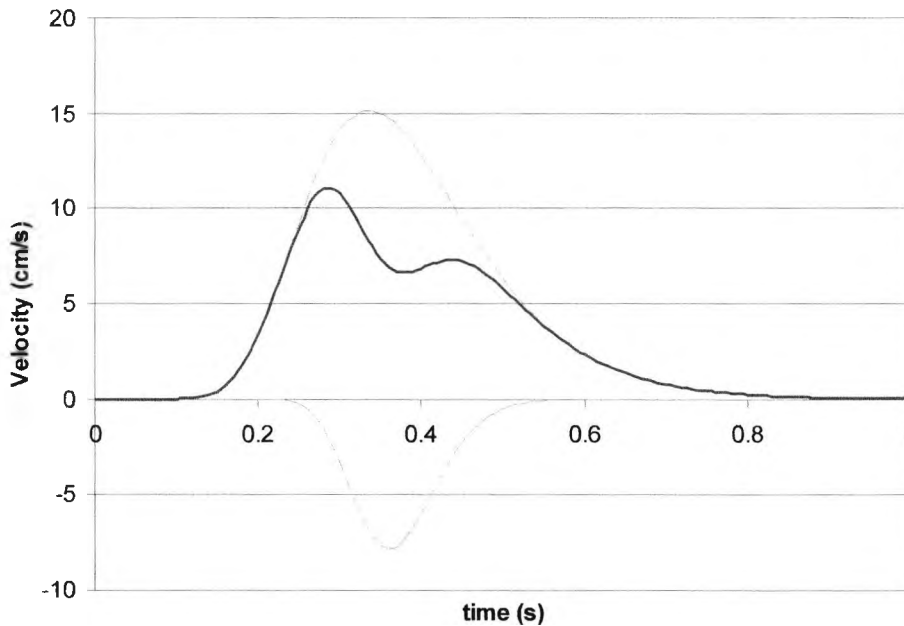


Figure 6.2 - Synergy of Agonist and Antagonist Systems

The synergy is composed of two parallel systems that represent, respectively, the sets of neural and muscular networks involved in the generation of agonist and antagonist activities which result in a specific movement. Each of these two systems can be considered as a linear time-invariant system producing a velocity output [  $V_{\sigma 1}(t)$  or  $V_{\sigma 2}(t)$  ] from an impulse command  $U_0(t - t_0)$  of amplitude  $D_1$  or  $D_2$  occurring at  $t_0$ , where the subscripts 1 and 2 stand for the agonist and the antagonist system, respectively.

Figure 6.3 shows a velocity curve produced, in a spreadsheet, by combining the output of two models. Figure 6.3 also shows the component curves produced by the agonist and antagonist models; these are shown as dashed lines, above and below the zero velocity axis respectively.

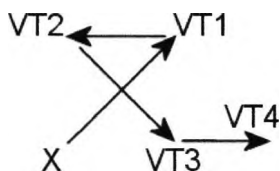


**Figure 6.3 - Modelled Velocity Curve from combined Agonist and Antagonist systems**

The velocity plots from lines captured from VMI test drawings are a lot more complicated. The overall velocity may be made up from many sets of movements. The arm, wrist, hand, fingers and even whole body may all contribute to the final velocity. Extracting the reconstruction parameters is not simple.

To confound matters, the test drawings are not produced by a single rapid-aimed movement. The subjects are copying a series of lines. More than one line may be made with a single stroke. Research by Adam et al. (2000) shows that control of a second movement may overlap with the execution of the first. This has to be taken into account in the analysis. Sometimes the subject will draw very slowly with the pen stopping several times along the same line. This will result in a large number of movements. For very small movements the sampling frequency of the tablet may not be great enough.

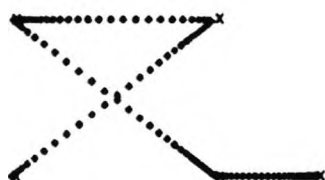
Plamondon and Privitera (1995) describe the process of producing a complicated movement, such as drawing or handwriting. The starts off as a set of planned targets; this is known as an engram.



**Figure 6.4 - Movement Engram**

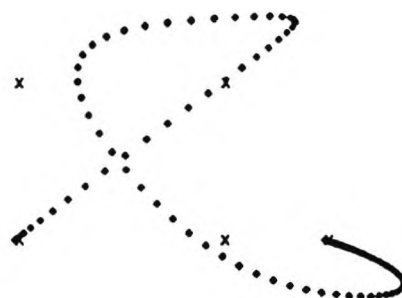
Figure 6.4 shows the movement engram for a simple shape which can be drawn as a series of connected pen strokes. The starting point of the engram is labelled X. Virtual Targets VT1 – VT4 are shown indicating the ends of each pen stroke.

Each of the strokes in the engram is performed as a rapid-aimed movement. There is a spatiotemporal trade off between the accuracy of the replication and the speed with which it is produced. This is due to the overlap of the movements. The faster the reproduction the further from the targets the line becomes.



**Figure 6.5 - Drawing with a low speed/accuracy**

(Plamondon and Privitera 1995)



**Figure 6.6 - Drawing with a high speed/accuracy**

(Plamondon and Privitera 1995)

Figure 6.5 and Figure 6.6 show drawing based on the engram, where each dot on the diagrams is equally spaced in time. Figure 6.5 has been drawn accurately as so matches the engram quite closely. Figure 6.6 has been performed faster and hence misses the Virtual Targets (shown as Xs) by a significant distance.

It is hoped that we will be able to extract the engram targets from the subjects drawings. The number of virtual targets and the speed/accuracy ratio of execution could be investigated as a potential feature for classification of Dyspraxia.

## **6.4 Filtering curves**

The velocity curves calculated from the VMI drawings contain a lot of noise. This can be seen to arise from many sources, some from the nature of the VMI test itself. For example, the friction of the pen on the paper or sampling inaccuracies of the graphics tablet. These all add up to a noisy set of samples which need to be filtered before they can be analysed.

Several filters were investigated. These included:

- Mean Averaging Filter on the Velocity (Various Sizes)
- Mean Averaging Filter on the Acceleration (Various Sizes)
- Reducing number of DCT coefficients of Velocity.
- Low Pass Butterworth Filter

The low pass Butterworth Filter was found to be the best at eliminating unwanted noise without destroying the features that we needed to analyse. A cut off frequency of 10Hz was used as suggested by MacKenzie et al. (1987).

## **6.5 Finding Appropriate Curves**

Velocity curves extracted from drawings of VMI shapes often have a large number of velocity peaks. They do not have a single bell shaped curve that we can match a Rapid Aimed Movement to. In order to fit a curve to the model we need to find lines, or parts of lines, which have a bell shape.

Brault and Plamondon (1993) have used corner detection to split a line at perceptually important points. Brucq et al. (1997) have also developed a segmentation system using detection of direction changes. As the lines we are using are straight, neither of these methods is suitable. The segmentation was done by visual inspection of the velocity curves.

To do this we had to search through all of our VMI drawing data to find any lines which could possibly be modelled by a single RAM. Software was written to assist with the search.

### 6.5.1 Software

A method of viewing the velocity curves was needed. To save time Matlab was used to view the curves. Curve data for all of our test patients was already contained in a single text file created by the VMI assist program (see Chapter 3). Getting the curve data into Matlab was achieved using a Visual C++ program. The program read in the drawing data from the file and computed the velocity data. The output of this program was another program which could be executed in the Matlab environment. Figure 6.7 shows the Visual C++ program in operation.

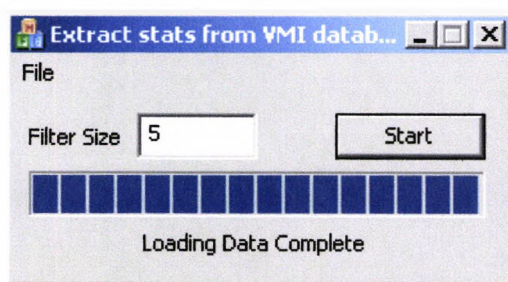


Figure 6.7 - Creating the Matlab code

As well as showing the raw velocity curves, a filtered curve was also displayed. Filtering was performed using a low-pass Butterworth filter with a 10Hz cut-off frequency (see Section 6.4). The Matlab program also labelled the graphs so the user could identify any appropriate curves for later analysis. Between displaying each curve Matlab would wait for a key press from the user. Figure 6.8 shows the unfiltered curve. Figure 6.9 shows the same curve after it has been filtered. Figure 6.10 shows an overlay of both the unfiltered and filtered curves.



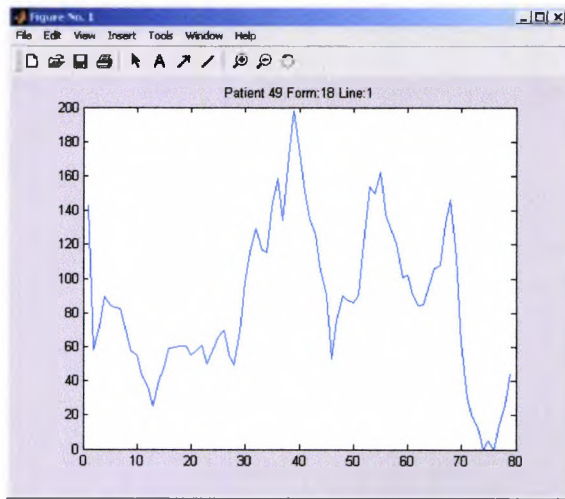


Figure 6.8 - Unfiltered Curve

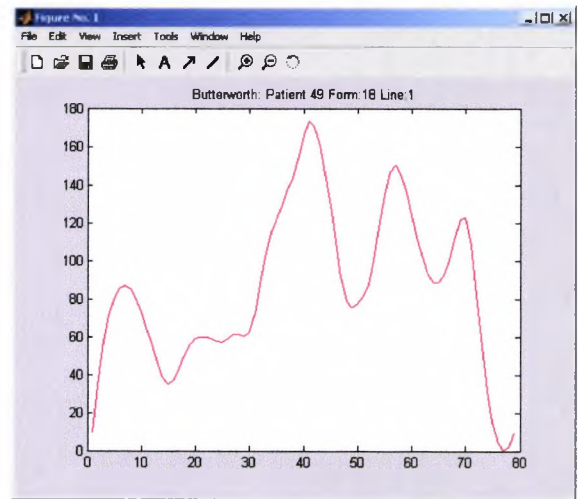


Figure 6.9 - Butterworth filtered curve

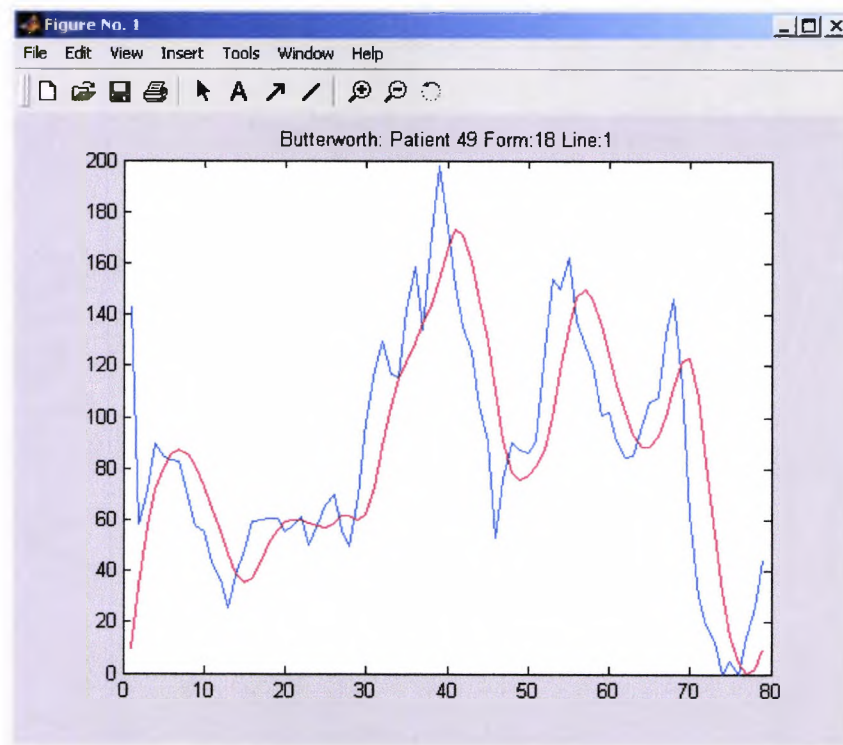


Figure 6.10 - Overlaid Curves

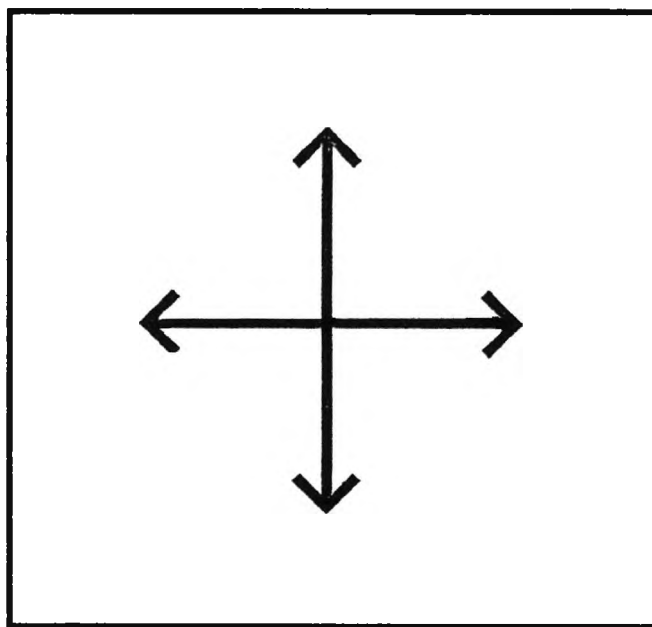
### 6.5.2 Curve Selection

The Matlab code was executed and the output observed. Each curve was inspected visually for possible RAM sections. When these were spotted a note was made of the patient number, shape number and line number so they could be analysed later. A note was also made with an estimate of how easy the curve would be to model.

113 suitable curves were identified as having potential for modelling.

- 46 good curves
- 47 reasonable curves
- 19 possible curves.

We decided to narrow down the selection of curves by splitting them depending on the shape that was being attempted. The shape with the greatest number of 'good' curves was shape 15 (see Figure 6.11).



**Figure 6.11 - Shape 15 from the VMI test**

There are 20 lines which contain good RAM sections and were found in shape 15.

### **6.5.3 Curve Extraction**

The lines identified in Section 6.5.2 were filtered using Matlab and then imported into Microsoft Excel.

The lines were trimmed so that only the section of interest remained. Some lines contained more than one section of interest so two sections were produced. Also some lines turned out to be not suitable for analysis.

The curve extraction produced 23 sets of data.

## **6.6 Parameter Estimation**

Now that we have a set of appropriate velocity profiles we can estimate the modelling parameters.

### **6.6.1 Software**

A C++ program was written for estimating RAM parameters. The software currently uses a British Library search algorithm which was fast to implement but requires extensive processing time to extract the parameters. Djeziri et al. (2000) suggest using Levenberg-Marquardt non-linear regression (Marquardt 1963). Dijoua and Plamondon (2005) developed a parameter estimation system using an evolutionary algorithm.

Figure 6.12 shows the user interface. The user can change the search range and step size of all of the search parameters.

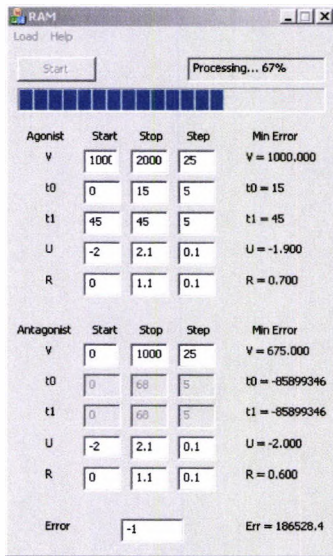


Figure 6.12 - Parameter Estimation

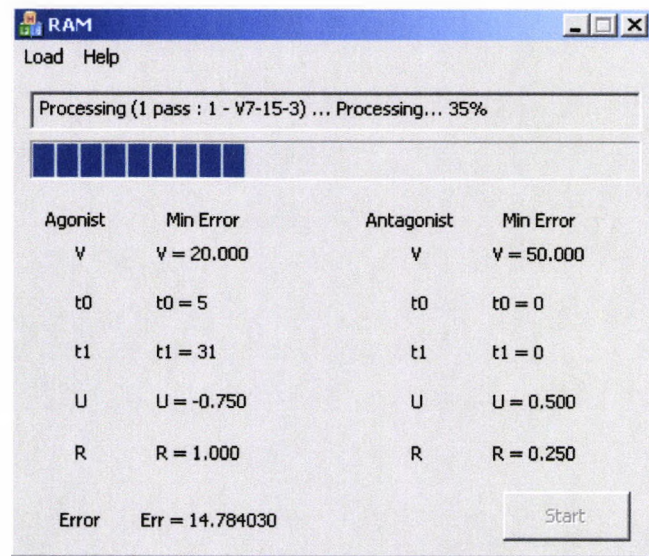


Figure 6.13 - Batch Processing

Due to the time taken to process each curve, the software was extended to allow a set of curves to be processed without supervision. An initial set of parameter ranges was hard coded into the software. After completing the search for the whole set of curves the parameter ranges would be narrowed and the step size decreased to close in on a local maxima. The results were saved to a file after processing each curve. The processing could therefore be resumed if anything happened to interrupt the software, for example, power failure or accidental closure of the software. Figure 6.13 shows this batch version of the software in operation.

### **6.6.2 Modelling**

The curve sections found in section 6.5 were fed into the batch estimation program (Section 6.6.1) and left to run overnight. This produced modelling parameters for the curves.

Microsoft Excel was used to create a modelled curve from the parameters found. A spreadsheet was created which would calculate a set of sample points for the modelled curve, these points could then be compared against the actual points from the line.

The results were copied into excel and graphs were plotted of all the curves. These graphs showed the original line, the modelled line, and also the agonist and antagonist components of the modelled line.

### **6.6.3 Analysis**

In a few cases there was a good match between the original and modelled curves. However, in the majority of cases the single RAM model was unable to reproduce the velocity curve.

The following notation is used for the plots in the remainder of this chapter:

- Solid Line – The original curve from the graphics tablet.
- Crossed Line – The modelled curve.
- Dashed Line – The component RAM curves (agonist and antagonist).

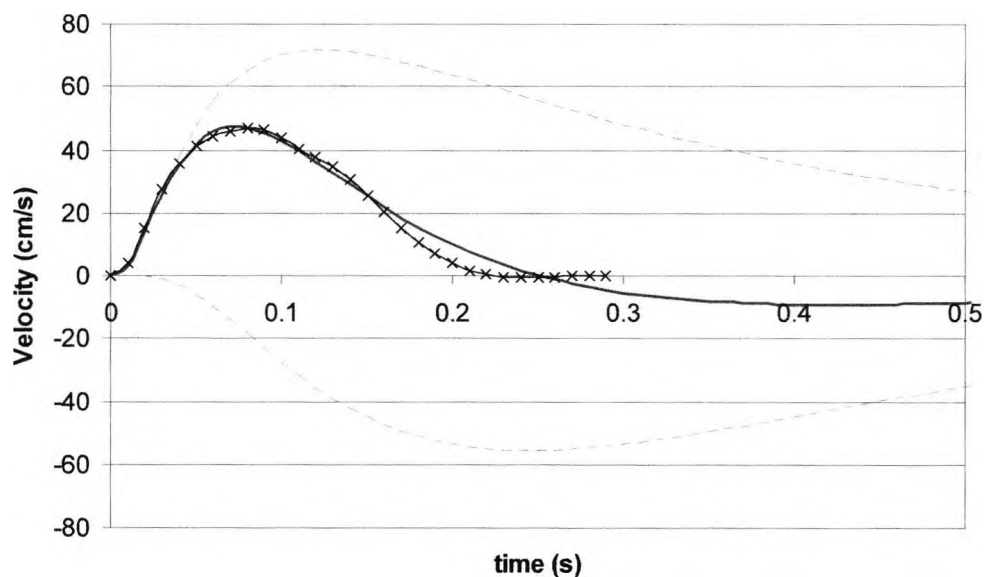


Figure 6.14 - Well modelled Control

Figure 6.14 and Figure 6.15 show curves generated which closely match the original curve. Even in these cases the model is not completely accurate. It appeared that even the simplest curves would need to be modelled with at least two RAM curves.

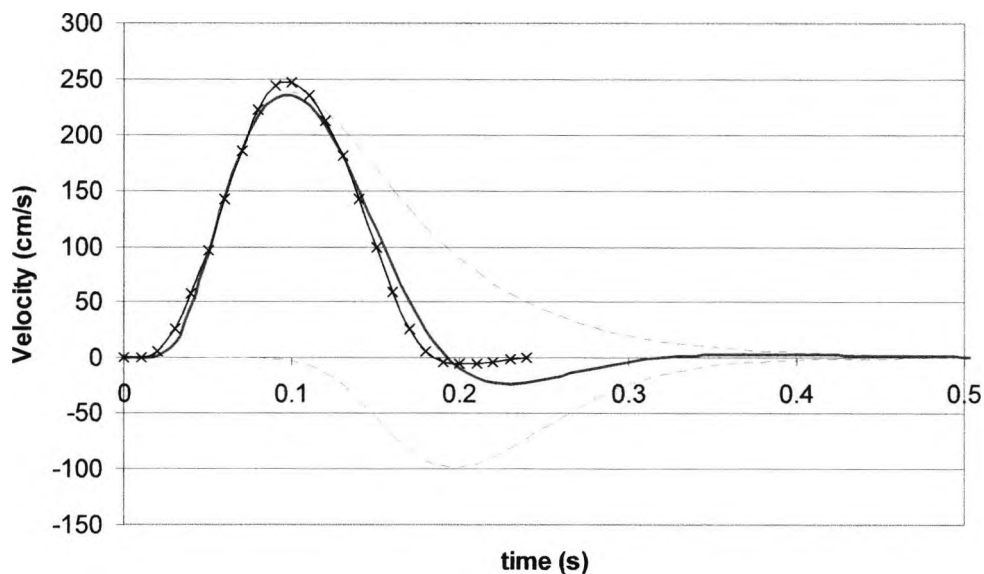


Figure 6.15 - Well modelled Patient

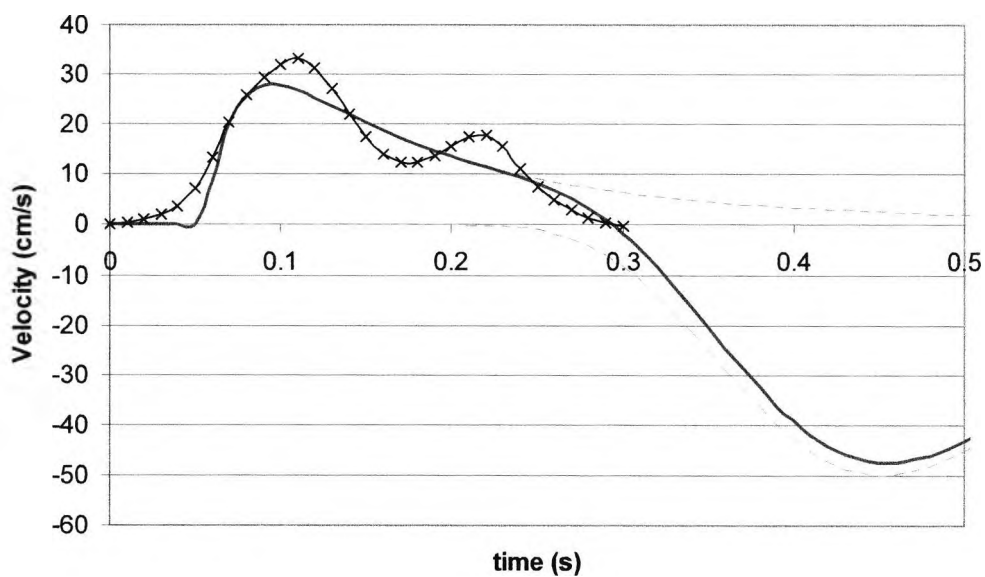


Figure 6.16 - Poorly modelled Control

Figure 6.15 and Figure 6.16 show two slightly more complex curves which the model fails to match accurately.

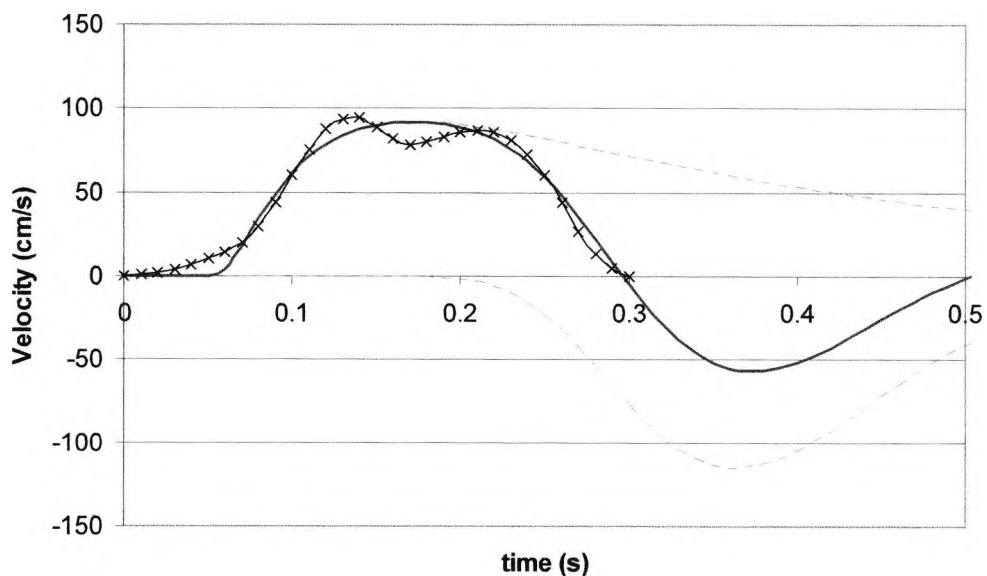


Figure 6.17 - Poorly modelled Patient

### 6.7 Multiple Rapid Aimed Movements

We have developed a process to model curves using more than one RAM. Several of the curves we have selected for modelling have two peaks, these seemed to be easiest to split into two bell shaped curves. Figure 6.18 shows the curve where the distinction between the two peaks was greatest. This Figure also shows the result of modelling with a single RAM. This curve will be used to describe the process.

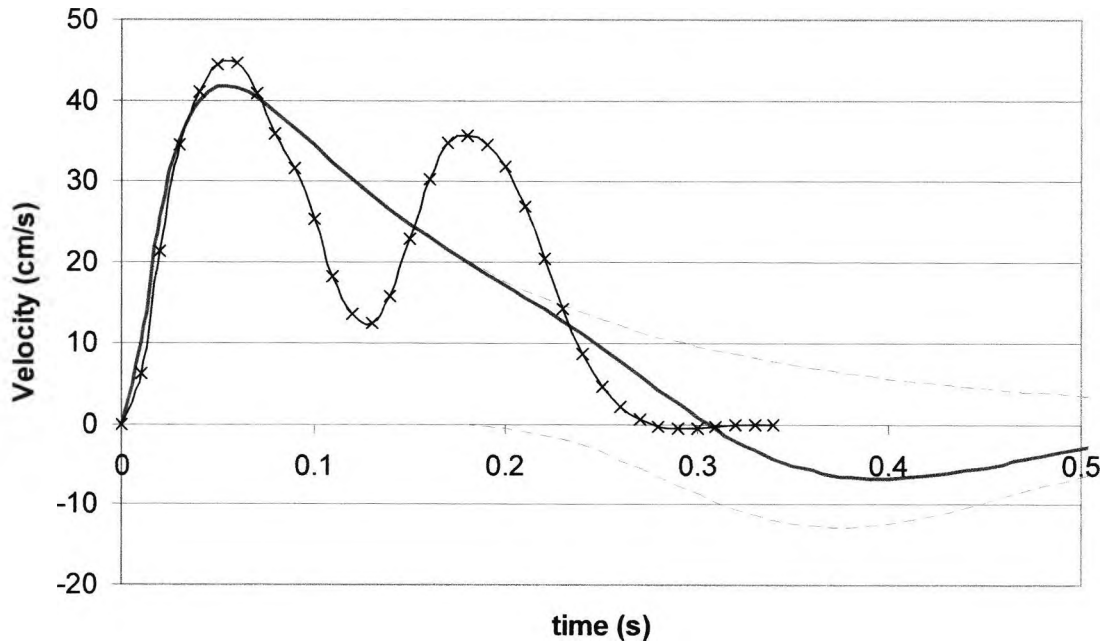


Figure 6.18 - Curve with two peaks

The curve was considered in two sections. The split between the two sections occurs at the local minima between the two peaks.

The first step is to estimate parameters which match a modelled curve to the first section of the curve. Estimating the parameters of just the first section produces good results for the section, but the model continues at high amplitude afterwards. Ideally this would be as low as possible so as not to interfere with the model of the second section. This was overcome by taking the entire curve and setting the samples after the split point to zero. The parameter estimation program then produces a more stable model of the curve. Figure 6.19 shows the model curve for the first section.



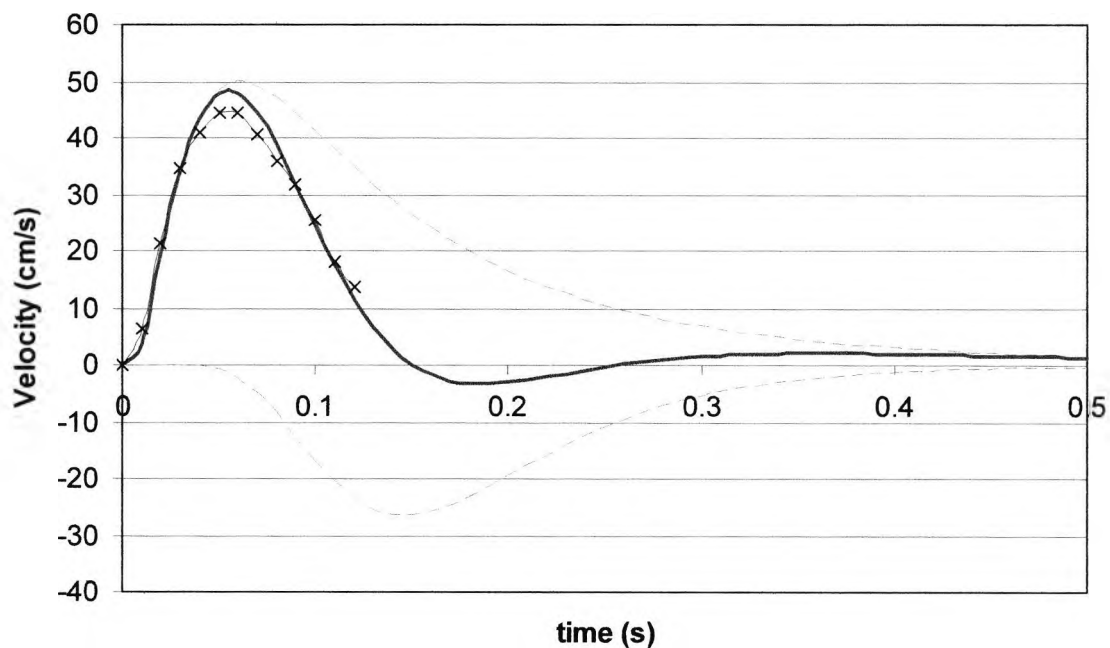


Figure 6.19 - Modelling section 1

The second section is created by subtracting the modelled curve of section 1 from the original full curve. This curve was fed into the parameter estimation program. Figure 6.20 shows the curve for section 2 and the modelled version.

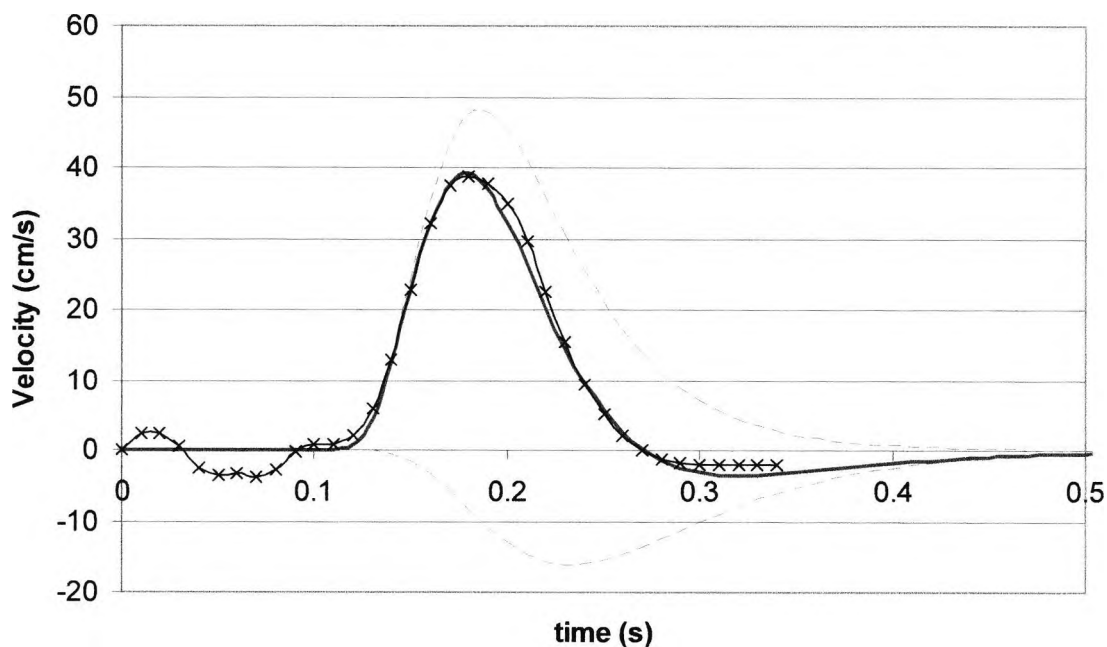


Figure 6.20 - Modelling section 2

We can now produce a model for the entire curve by adding together the curves produced by the two models. Figure 6.21 shows the original curve along with the model for the entire curve and the models for section 1 and 2.

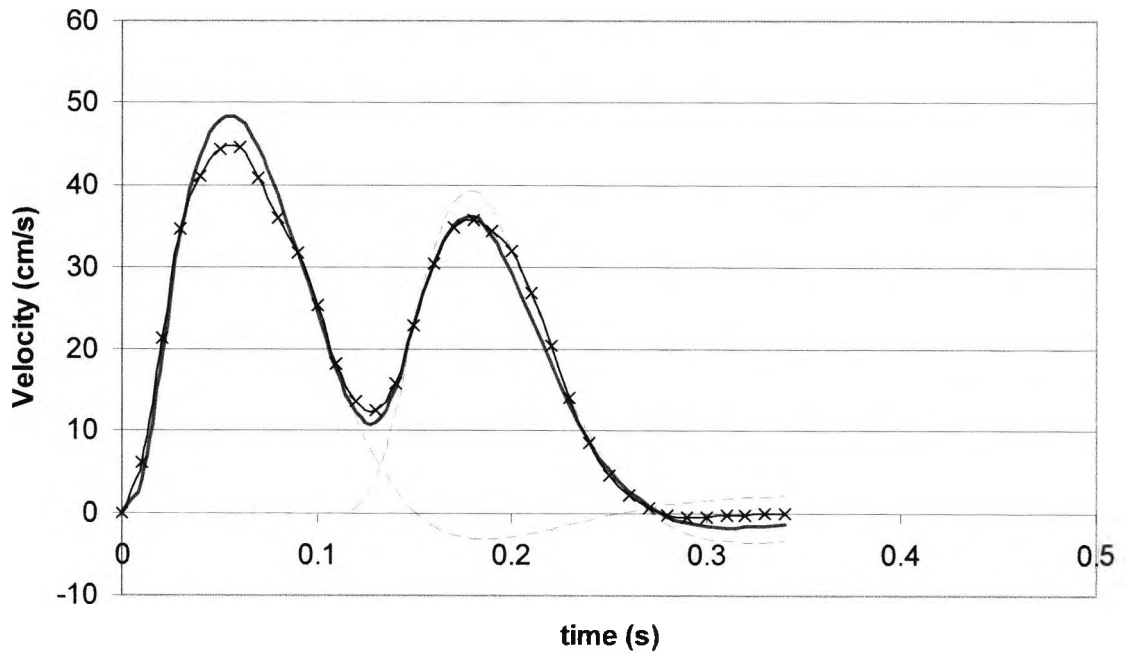


Figure 6.21 - Initial Full Model

To further improve accuracy a second model of the first section was produced. The desired curve was created by subtracting the modelled curve of section 2 from the original full curve. The parameter estimation program was used to produce the model. Figure 6.22 shows the curve produced by the improved model of section 1.

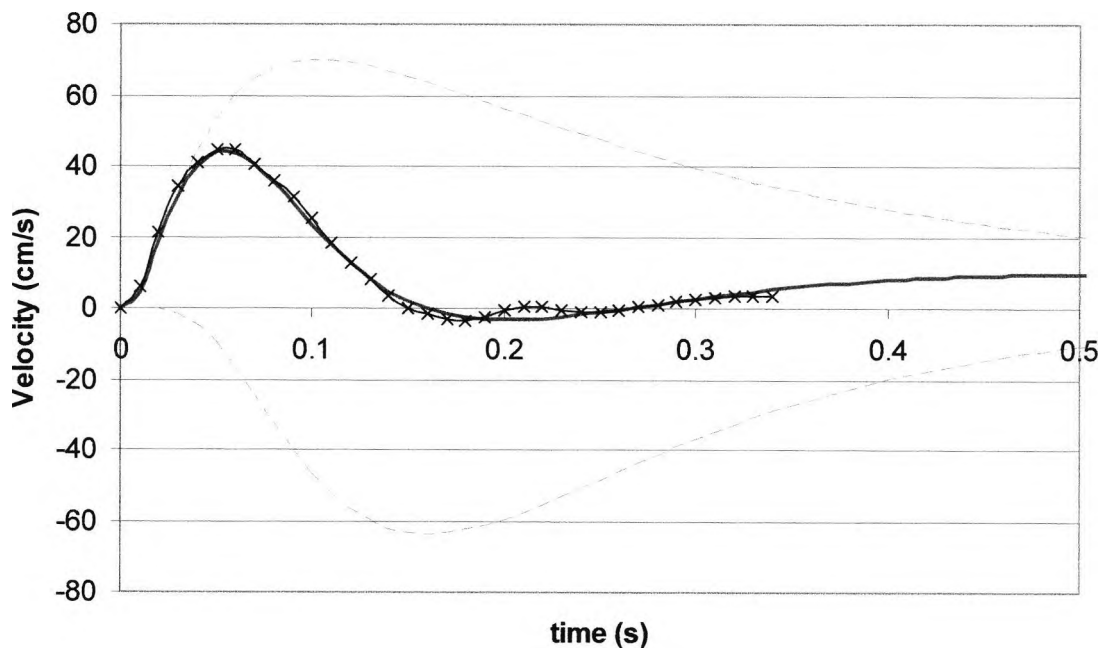


Figure 6.22 - Remodelling Section 1

The curves produced by modelling section 2 and re-modelling section 1 were then combined to produce a more accurate model for the entire curve. Figure 6.23 shows the model for entire RAM.

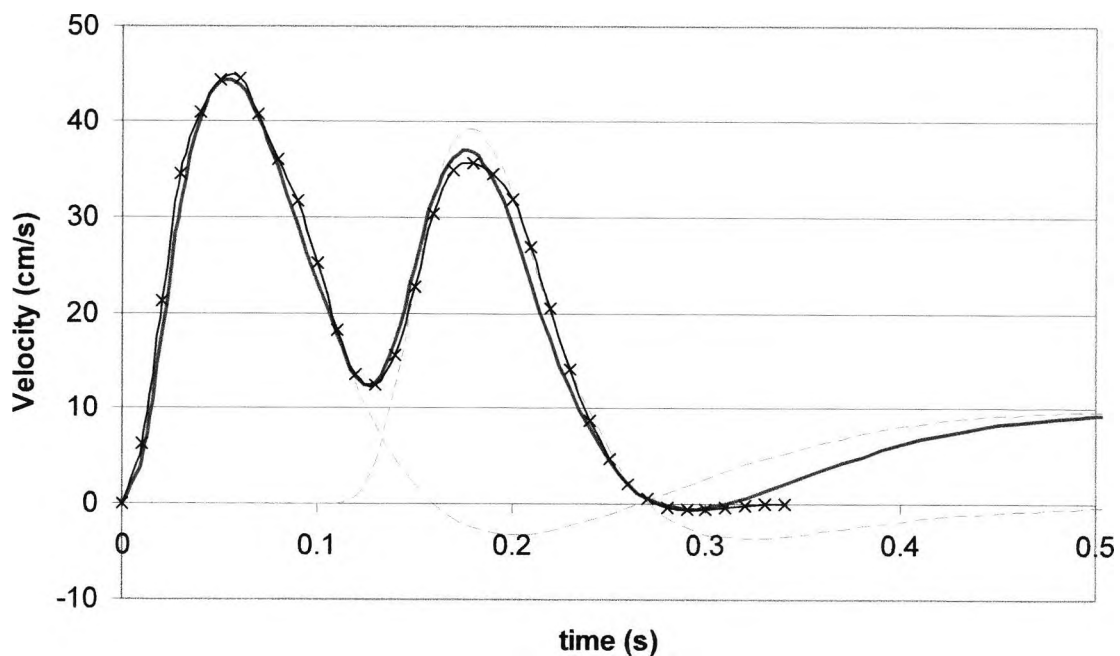


Figure 6.23 - Modelled Complete Curve

Figure 6.24 shows a flow diagram of the steps that need to be performed to produce a model for a pen stroke consisting of 2 rapid-aimed movements. If the initial full curve model is accurate enough, the remodelling of section 1 can be omitted. This saves a lot of time as the parameter estimation program only needs to be run twice instead of three times.

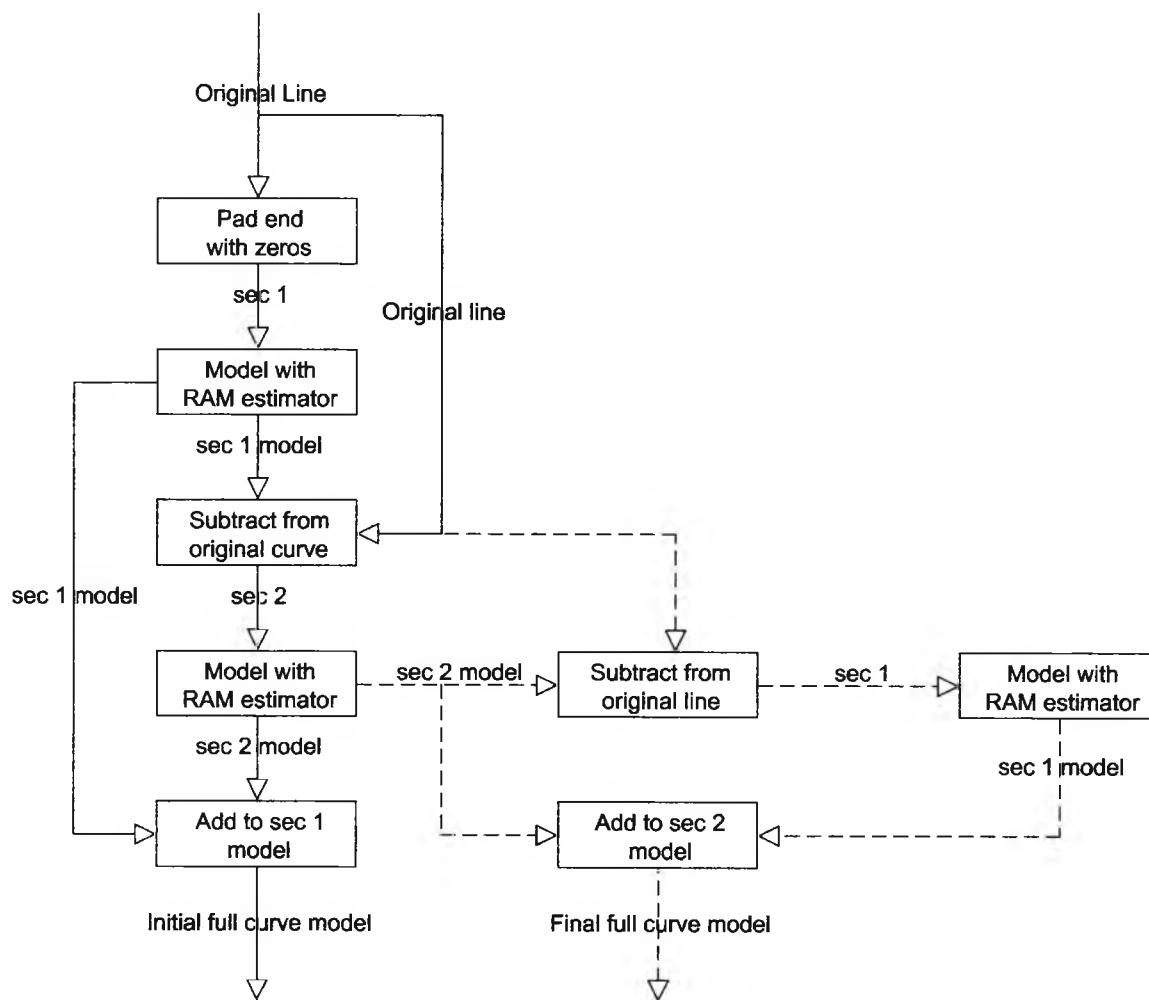


Figure 6.24 - Modelling Flow Diagram

### 6.8 Modelling Shape 15 curves

The curve modelling process was applied to the set of curves found from shape 15. There is no obvious discrimination between the control and patient parameters in this sample. Even parameters from the same subjects are not consistent across different lines.

Figure 6.25 and Figure 6.26 show an example of the distribution of parameters for the set of curves. Dyspraxic patients are shown as pink squares. Control subjects are shown as blue diamonds.

In the plots for other pairs of parameters, the Patient and Control curves are equally inseparable.

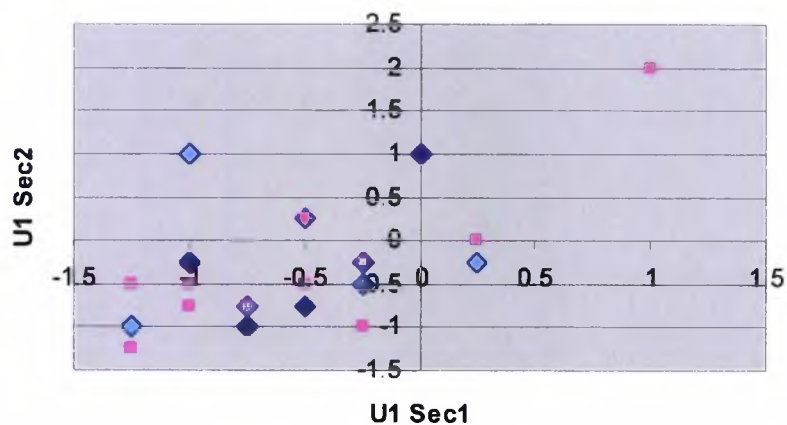


Figure 6.25 - Comparing U1

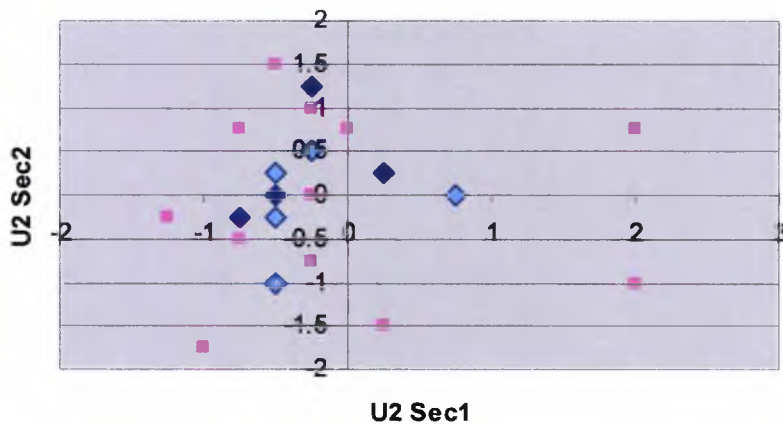


Figure 6.26 - Comparing U2

### 6.9 Modelling Shape 15 Arrowheads

A more thorough analysis of shape 15 was undertaken to look more closely at the RAM strokes. The arrowheads of the shape are short lines so these are where RAMs are most likely to be found. The direction of the stroke is important as the pen would be controlled by different muscle groups.

At first we considered only the 10 year old subjects. It was hoped that several lines would be found which consisted of only one Rapid Aimed Movement.

We sorted the lines into the direction of the stroke. The majority of the arrowheads were drawn with a single line so needed to be split into two sections. Once the sort was completed we had four sets of lines:

- Up-left
- Down-left
- Up-right
- Down-right

The down-right direction was found to be used significantly more than the other directions. For this reason this direction was used for the analysis. From this set only 3 suitable lines were found. These were all from patients. The analysis was extended to 6 year olds. These should provide the greatest difference in motor control performance. From this set only 5 suitable lines were found, these were all from control subjects.

Figure 6.27 shows the velocity plots of the suitable curves extracted from the arrowhead of shape 15. Plots from Dyspraxic patients are shown in blue, control subject's plots in red.

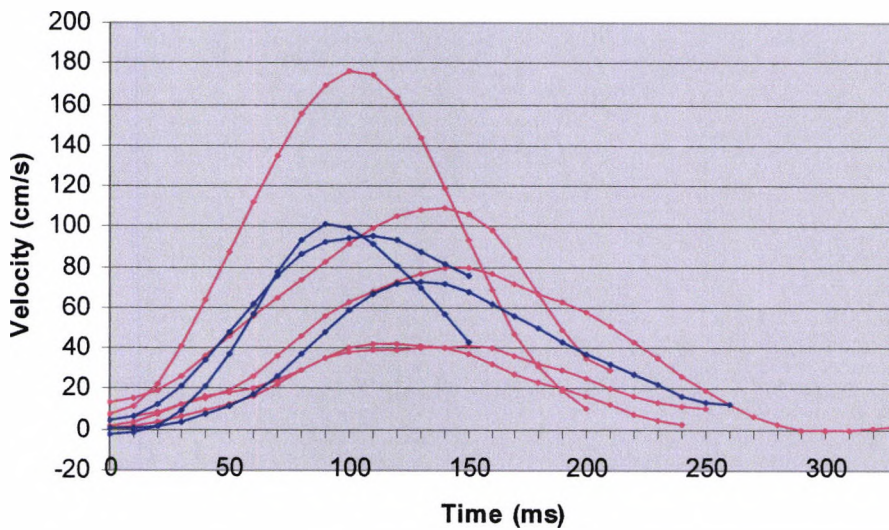


Figure 6.27 - Velocity curve plots for arrowheads of shape 15

The RAM parameters that were extracted from these curves are shown in Table 10. T0 and T1 are the start and end times of the modelled movement. V, U and R are parameters of the system. V is the magnitude of the impulse command. U is the time delay of the system and R is the response time.

Parameter	Control Subjects (6 year old)					Dyspraxic Patients (10 year old)		
T0 (x10 ms)	0	0	0	0	0	0	0	0
T1 (x10 ms)	21	21	31	21	21	31	11	16
Agonist V	130	80	120	200	120	50	200	180
Agonist U	0.75	0.25	0	0.25	-0.5	-0.5	0.25	0.25
Agonist R	1	0.75	0.75	0.75	0.5	0.5	0.5	0.75
Antagonist V	30	20	40	80	40	10	190	190
Antagonist U	0.25	0.25	0	0.25	0	-0.25	0.75	1
Antagonist R	0.25	0.25	0.25	0.25	0.25	0.25	0.5	0.5

**Table 10 – RAM parameters from arrowheads of shape 15.**

From these results we again see that there is no obvious pattern which can be attributed to more highly developed motor control. The parameters are scattered with no real grouping of any kind. This may be due to the small sample size, or the unrestrained nature of the drawing task.



## **6.10 Conclusion**

We have developed software to estimate the parameters for a mathematical model of a single Rapid-Aimed movement. We have created a process to extend this to model more complex curves.

The curves extracted from the VMI test were much too complex to be dealt with in their entirety. Software was therefore written to find sections of the velocity curves that looked suitable for modelling.

Small sections of the curves were analysed. However, these failed to produce any insight into the differences between Dyspraxic and Control drawings. This may be because the VMI drawing are produced with a slower controlled pen movement rather than a rapid-aimed movement.

For future work it would be valuable to isolate RAMs using a specifically designed drawing task. This would remove a lot of the variability from the experiment and perhaps provide a more meaningful set of results. However, our conclusion from the initial analysis described here is that the concept of Rapid-Aimed Movements is not one which can readily carry over to a direct analysis of drawings produced in the VMI test. Thus, an alternative strategy for dynamically analysing the drawings produced in this type of test must be sought.

Although Rapid-Aimed Movements may not be suitable for analysing drawings from the VMI test, an alternative to specifically developing a dynamic movement-modelling capability may be to look more broadly at the drawing characteristics which might be extracted from the drawings produced. The next chapter looks at some more general measurements and how these might be more revealing in developing a more objective characterisation and understanding of the performance of Dyspraxic children.

# Chapter 7

## Dynamic Shape Analysis

### **7.1 Introduction**

Modelling parameters of RAMs taken from the VMI test has proven to be not very useful, due to the rarity of such a movement and the variability in movement strategy. The theory that all strokes are made up of a series of RAMs is still of interest though. In a straight line the direction of the component RAMs will be consistent but the number and size of RAMs will vary.

A study was undertaken into the number of RAMs used to generate straight lines while performing the VMI test. Smits-Engelsman (2001) suggests that Dyspraxic children will have fewer velocity peaks. This is because they prefer a more ballistic movement strategy that is less dependent on visual correction.

The study was then extended to include more quantitative metrics. Cousins and Smyth (2003) indicate that measures of movement time may differentiate between children with and without Dyspraxia. Software was written to extract these measures automatically.

Feature extraction was performed on shapes with unconnected straight line sections. We then looked at more complex shapes to try to detect differences in movement strategy.

Rémi, Frélicot and Courtellemont (2002) have used the drawing sequence as a feature to discriminate children with learning difficulties. Shimaya (1997) shows that a complex shape is perceived differently depending on ability. A sample shape (see Figure 7.1) is open to several interpretations (see Figure 7.2, Figure 7.3 and Figure 7.4).

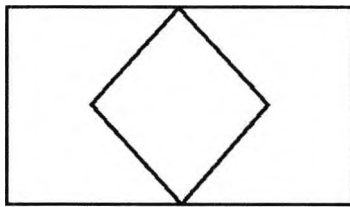


Figure 7.1 - Sample Shape

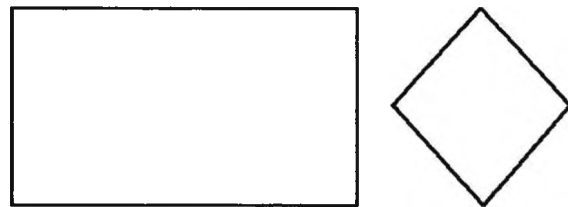


Figure 7.2 - Interpretation 1

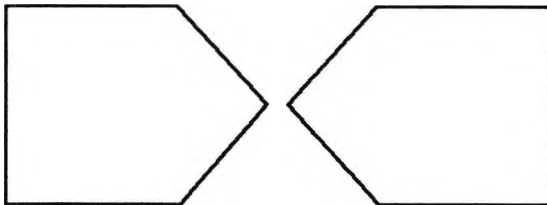


Figure 7.3 - Interpretation 2

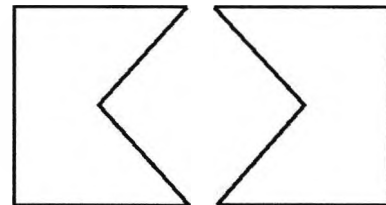


Figure 7.4 - Interpretation 3

Shapes were compared which contained a similar subset of lines.

## 7.2 Counting rapid-aimed movements

Following up the research in chapter 6, the number of RAMs that made up a line was of interest to us.

The arrowheads from shape 15 were used for our comparison. Estimating the number of RAMs was performed manually so shorter lines provided a more reliable result. The count is based mainly upon the number of velocity peaks in the curve but also takes into account any irregularities in the sides of the peaks.

Only one age group was used for this initial comparison. 10 year old subjects were chosen as they are more likely to have completed the shape successfully. Beery (1997) notes that producing sharp points on the arrowheads is particularly valuable as an indicator of a young child's developing directionality.

Patients were excluded from the analysis if the arrowheads were not drawn as separate lines to the cross, or if they failed to make sharp points.

Table 11 shows the average number of RAMs used per stroke. The control subjects use more RAMs during the production of the arrow points. This agrees with Smits-Engelsman (2001).

Subject	Average RAMs per stroke
Control A	3.38
Control B	3.63
Control C	3.57
Patient A	2
Patient B	1.25
Patient C	2.75

**Table 11 - Average number of RAMs**

This method requires visual inspection of the curves by a person. We continued the investigation by looking for measurements that could be taken objectively.

### 7.3 Software Analysis

In order to expand the analysis, Software was created to compute metrics from the velocity information.

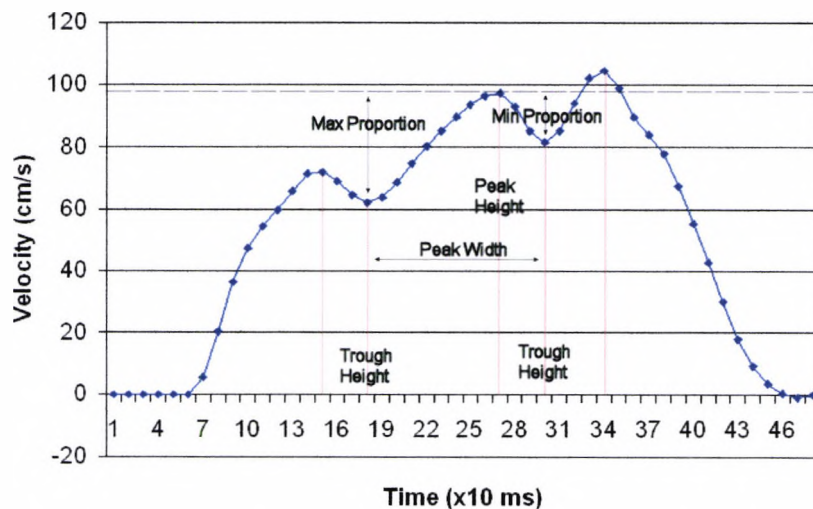


Figure 7.5 - Measurements from the velocity curve

Figure 7.5 shows a velocity curve which has been labelled to indicate the measurements which were taken. The first step is to find the locations and velocities of the peaks and troughs. These are identified by stepping through each sample value looking at the neighbouring points to determine if it is a local maxima or minima. The remaining metrics are then calculated from these.

The 'Proportion' measures are the height that the peak reaches from the neighbouring troughs. The 'Max Proportion' is the side with the greatest drop and the 'Min Proportion' the smaller drop. The percentage proportion is the measurement as a percentage of the peak height.

The features extracted were:

- Num Points
- Num Peaks/Troughs
- Width of peaks
- Height of peaks
- Height of troughs
- Max Proportion
- Max % Proportion
- Min Proportion
- Min % Proportion

For each measure the minimum value, maximum value and the mean average for all peaks was calculated.

The lines making up the arrowheads of shape15 were analysed using the software. The most prevalent direction of the component strokes was down-right. Only these strokes were considered. We looked at drawings from 6 year olds and 10 year olds. These are the upper and lower age groups so should show the biggest difference in performance.

Figure 7.6 shows a plot of two of the metrics for our test set.

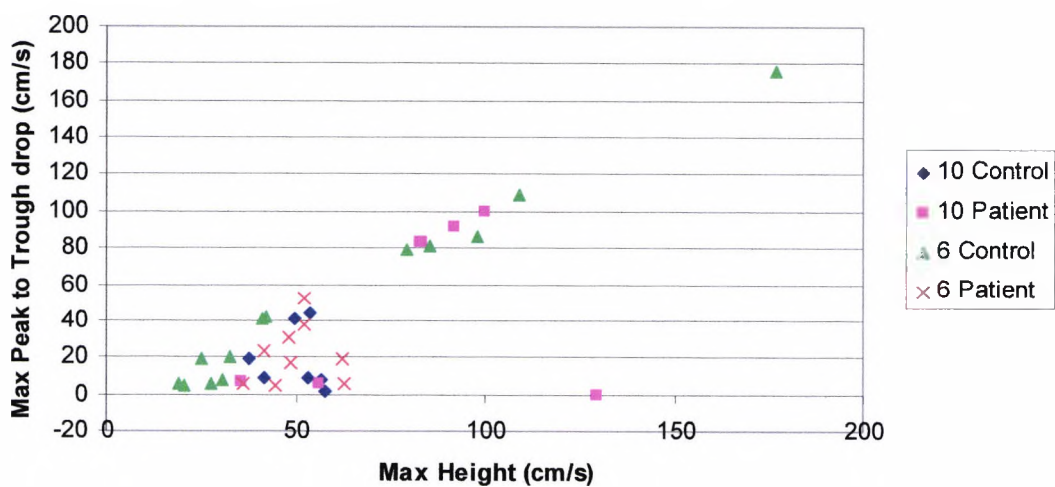


Figure 7.6 - Max Velocity against Max Proportion

The highest velocity peak seems to separate out the controls from the patients. This value is also the greatest overall velocity for the curve. The other metrics seem to either vary with the max height or are of no use in splitting the groups. It is interesting to see that the 6 year patients and 10 year controls are very closely matched.

#### 7.4 *Straight line sections*

To compare how the subjects drew similar lines under different conditions we have looked at shapes which included vertical and horizontal lines which are straight and unconnected. Figure 7.7 to Figure 7.10 show the four shapes which contain suitable lines.

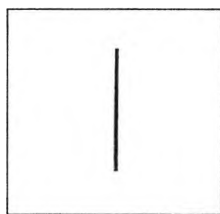


Figure 7.7 – Vertical  
Line

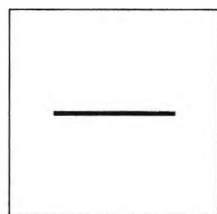


Figure 7.8 –  
Horizontal Line

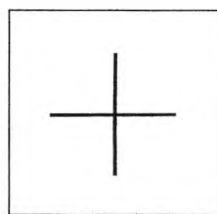


Figure 7.9 – Vertical-  
Horizontal Cross

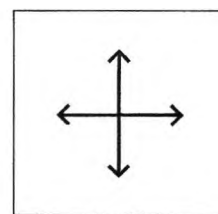


Figure 7.10 –  
Directional Arrows

The vertical-horizontal cross (see Figure 7.9) was split into one vertical and one horizontal line. The arrowheads from the directional arrows shape (see Figure 7.10) were removed and the vertical and horizontal lines separated. The vertical line and horizontal line (see Figure 7.7 and Figure 7.8 respectively) required no extra manipulation.

We initially restricted the analysis to only one age group to try to create a defined set of results that were not influenced by extra factor such as age variation. We chose to use the oldest subjects, the 10 year olds. The analysis was then extended to the youngest subjects (6 year olds) to see the progression of each group over time.

### 7.4.1 Features

The filtered curves were viewed and the number of RAMs needed to model the curve was estimated. The feature extraction program was also used to calculate objective measurements.

### 7.4.2 Analysis

We found that the most useful of the metrics were the number of RAMs and the number of samples. The number of samples is directly related to the drawing time.

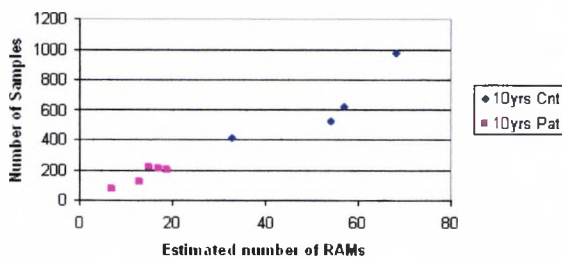


Figure 7.11 - 10 year olds Vertical Line

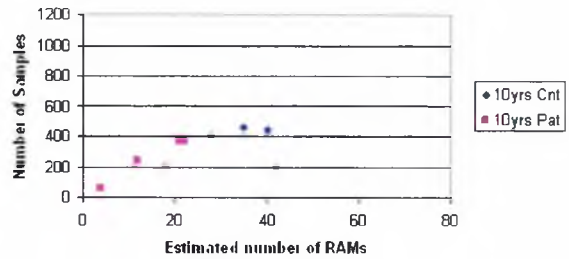


Figure 7.12 - 10 year olds Horizontal Line

Figure 7.11 and Figure 7.12 show the time taken to draw the line plotted against the number of RAMs for shapes 4 and 5, respectively, from the VMI test, for 10 year old children. The number of RAMs separates the control group from the Dyspraxic group. It can be seen that the number of RAMs increases with the number of samples.

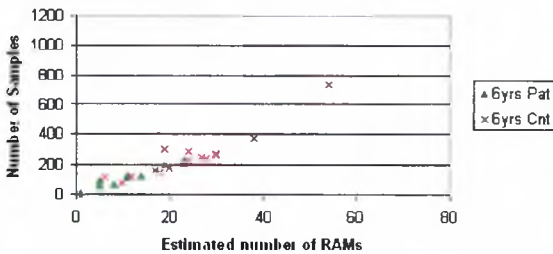


Figure 7.13 - 6 year olds Vertical Line

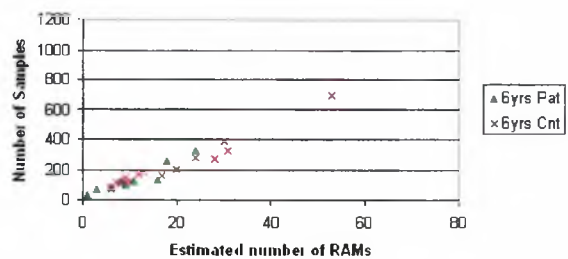


Figure 7.14 - 6 year olds Horizontal Line



Figure 7.13 and Figure 7.14 show the same values plotted for 6 year old children. There is a lot of overlap between the control and patient groups, but there does seem to be a tendency for Dyspraxic children to draw with fewer RAMs and in less time. This is less pronounced than in the 10 year old children.

### **7.4.3 Conclusion**

The Control subjects overall seem to spend longer drawing the line than the patients, also the number of RAMs estimated is also greater. However there is significant overlap of the two data sets at the lower left sections of both graphs. The ratio of samples to number of RAMs appears to be fairly consistent across all of the subjects. In the horizontal lines there is a slightly greater number of RAMs per sample for the cross shape compared to the single line, but this is consistent for both patients and controls.

The difference in the velocity profiles of the controls and patients seem to lie mainly in the speed at which the line is drawn. This is displayed in the time taken to complete the line and in the maximum velocity attained. The difference is not as great in 6year old subjects with the results overlapping substantially, in 10 year olds however there is a much more significant variation. This is a problem as the Dyspraxia needs to be diagnosed at the earliest possible stage.

These results indicate that Dyspraxia may affect the planning of the strokes rather than the motor activities. This suggests that it is a higher level disorder. We need to test the cognitive abilities of the children in order to confirm these conclusions.

### 7.5 Cognitive Analysis

To test the cognitive functions of the patients and controls, we decided to look at the execution of the entire shape not just single pen movements.

#### 7.5.1 Square (VMI shape 9)

The square was chosen as it allows comparison between pairs of parallel lines. One drawn unbounded, and one aimed to join up with a target.

The order in which the lines were drawn was deemed important as this allows us to illicit the drawing strategy of the children. Table 12 shows the different strategies which were used and the percentage of children that used each one.

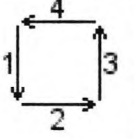
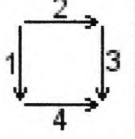
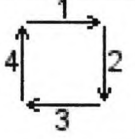
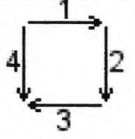
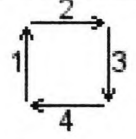
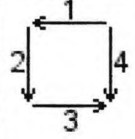
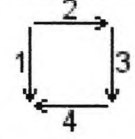
Strategy							
Control	7 (44%)	3 (19%)	1 (6%)	1 (6%)	1 (6%)	1 (6%)	2 (13%)
Patient	5 (56%)	0 (0%)	3 (33%)	0 (0%)	0 (0%)	0 (0%)	1 (11%)

Table 12 - Drawing strategies for VMI Square

The number of times pen strokes was also measured. The end of a pen stroke was defined as when the pen left the paper. A shape drawn with one continuous line would count as one stroke, even if the pen stopped at the corners. Table 13 shows the percentage of children that used each number of pen strokes. 5 was the maximum number used. An extra line is sometimes used to correct any gaps in the square.

Pen Strokes	Control	Patient
1	8 (50%)	8 (89%)
2	2 (13%)	1 (11%)
3	2 (13%)	0 (0%)
4	2 (13%)	0 (0%)
5	2 (13%)	0 (0%)

**Table 13 - Number of pen strokes for VMI Square**

Further analysis was performed on the subjects that had drawn the square in the most prevalent order. All the Patients that had used this order had used 1 pen stroke. All but one of the Controls has also used one pen stroke. The Control that had used two strokes was discarded. Also two further control subjects had made corrections to the shape after finishing the fourth side. These were also discarded.

We are left with 4 controls and 5 patients that have drawn the shape in the same way, with one line, in the same order, and with no corrections.

The mean velocity, standard deviation of velocity and number of samples in stroke (time) were computed for each of the sides of the square.

The average measurements for the controls and patients were taken and are show in Table 14 to Table 17.

Side	1	2	3	4
Control	1870	2002	1967	1817
Patient	734	746	572	786

**Table 14 - Time taken to draw each side (ms)**

Side	1	2	3	4
Control	24%	26%	26%	24%
Patient	27%	27%	19%	28%

**Table 15 - Percentage of time spent on each side**

Side	1	2	3	4
Control	1.30	0.85	1.09	0.91
Patient	3.54	3.30	4.93	3.14

**Table 16 - Standard deviation of velocity (cm/s)**

Side	1	2	3	4
Control	2.63	2.30	2.37	2.22
Patient	5.60	5.11	8.04	6.03

**Table 17 - Mean velocity of pen (cm/s)**

The Dyspraxic children draw the shape faster than control subjects. The shorter time taken to draw each side and the higher mean velocity bear this out. The standard deviation of velocity is also a lot higher for Dyspraxic children.

### **7.5.2 Open Square and Circle (VMI shape 13)**

A second shape was chosen to continue the analysis. Shape 13 of the VMI, the Open Square and Circle, was used. The open square can be compared with the results for the Square from section 7.5.1.

Table 18 shows the drawing strategies that were used for this shape and the percentage of children that used each one.

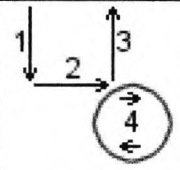
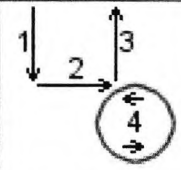
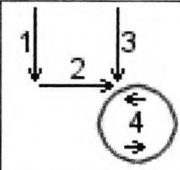
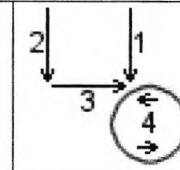
Order				
Control	3 (27%)	8 (73%)	0 (0%)	0 (0%)
Patient	0 (0%)	6 (75%)	1 (13%)	1 (13%)

Table 18 - Drawing strategies for VMI Open Square and Circle

Table 19 shows the number of pen strokes which were used to draw this shape.

Strokes	Control	Patient
2	10 (91%)	2 (25%)
3	0 (0%)	4 (50%)
4	1 (9%)	1 (13%)
5	0 (0%)	1 (13%)

Table 19 - Number of pen strokes for VMI Open Square and Circle

Again matching drawing strategies from control and patient populations were compared more thoroughly. Table 20 to Table 23 shows the metrics for this shape.

Side	1	2	3	4	Whole
Control	1210	2260	1850	4590	9900
Patient	920	760	1260	2030	4980

Table 20 - Time taken to draw each side (ms)

Side	1	2	3	4
Control	13%	22%	19%	46%
Patient	19%	15%	25%	41%

Table 21 - Percentage of time spent on each side

Side	1	2	3	4	Whole
Control	1.69	1.03	1.29	1.20	1.47
Patient	2.80	2.94	3.02	2.09	2.70

**Table 22 - Standard deviation of velocity (cm/s)**

Side	1	2	3	4	Whole
Control	2.48	1.33	1.47	2.25	1.60
Patient	3.92	4.05	3.53	4.09	3.85

**Table 23 - Mean velocity of pen (cm/s)**

As with the square we see that as the Dyspraxic child draws the shape more quickly, the time taken for each side is consistently less and the mean velocity for each side is higher. The standard deviation of velocity is again much higher for Dyspraxic children than control subjects.

### 7.5.3 Comparing Shapes 9 and 13

We compared the lower three lines of the square with the open square section of shape 13. These sub-sections are the same in both shapes.

First we found subjects that had drawn the three lines in the same way. They needed to have used the same number of strokes and the same order of lines.

In Controls there was very little overlap between drawing strategies of the two shapes. Only one control subject matched these criteria. Seven patients however were available. The results from the patients were averaged to produce a single value to compare. Table 24 to Table 27 show the results for this comparison.

Side	1	2	3
Control Square	1780	2210	2280
Control Shape 13	1240	2850	1930
Patient Square	1356	1077	1220
Patient Shape 13	914	761	1200

**Table 24 - Time taken to draw each side (ms)**

Side	1	2	3
Control Square	28%	35%	36%
Control Shape 13	21%	47%	32%
Patient Square	36%	30%	34%
Patient Shape 13	32%	27%	41%

**Table 25 - Percentage of time spent on each side**

Side	1	2	3
Control Square	1.51	1.79	1.40
Control Shape 13	1.59	0.76	1.22
Patient Square	2.23	2.74	2.84
Patient Shape 13	2.74	2.79	2.94

**Table 26 - Standard deviation of velocity (cm/s)**

Side	1	2	3
Control Square	3.22	2.87	2.74
Control Shape 13	1.68	0.84	1.17
Patient Square	4.48	5.36	4.84
Patient Shape 13	3.71	3.83	3.59

**Table 27 - Mean velocity of pen (cm/s)**

## **7.6 Conclusions**

The work so far suggests that the patients draw the figure faster and with less pen strokes than the controls. The standard deviation measured is always much greater for the patients than for the controls. There is also more variability of strategy for the control subjects. This may be because using more pen strokes allows for a greater number of possibilities.

The metrics which we found to be the best at separating Dyspraxic children from control subjects were the time taken to draw the line, and, standard deviation of the velocity curve.



# Chapter 8

## Conclusion

### ***8.1 Overall Research Conclusion***

The main objective of this work has been to investigate improvements to the diagnosis of Dyspraxia using novel computer tools. A computerised step added into a standard pen and paper test was used as the starting point for this research.

The direction of this work involved developing a computer tool to assist a clinician to perform analysis of children's drawings based on the Beery Developmental Test of Visual-Motor Integration (VMI). An analytical tool was then developed with a more general aim of the automatic assessment of any drawn shape, given a set of specific objective criteria. A set of templates was created with which the general scoring software could assess the VMI test. The research was also extended to look at the data produced during the VMI test in ways which would not be possible from the pen and paper test alone. A summary of the achievements and conclusions drawn from this research work are presented here.

Chapter 2 described the VMI test and how it has been extended to gather data from the children's drawings. Data collection was performed at a local State Primary School to bolster data already available within the department. The addition of a graphics tablet to the standard test was found to be non-intrusive. In fact the children

were intrigued by the computer and very receptive to the modified test. The set of existing software tools used is also described.

Chapter 3 introduces a software system which was developed to assist a clinician to score the VMI test. We have discussed some of the benefits of including a computer processing step into a standard assessment test. Measurement tools and real time playback of the drawing help a scorer to reach a more objective decision. Switching to a computer based analysis from a paper based one has the advantage of portability. Tests performed in a school can be sent to a clinician much more efficiently than a huge pile of test booklets.

Chapter 4 takes software scoring to the next level with a system designed to automatically assess hand drawn shapes with minimal human interaction. This software has been designed with the aim of providing a framework which can perform verification of any copied figure against a set of formally specified criteria. We have shown how geometrically important features of a drawing can be extracted and subsequently matched to a reference modal. A framework is provided around which a set of further verification rules can be implemented, specific to a particular shape. The use of algorithmically defined scoring rules results in a highly consistent method of evaluation.

Chapter 5 described a set of verification rules which were designed to assess the VMI test. The scoring accuracy of each shape has been compared to human scoring. A high correlation was found for the majority of shapes. An average scoring accuracy of 87% was found across all shapes. We have shown that the computerised system scores more harshly than a human. When a scoring mismatch occurs it is more likely to be where the human has scored a pass and the computer has scored a fail. As part of an initial screening program this would be effective for narrowing down a large group of children into a smaller set of candidates for further assessment.

Chapter 6 moves our analysis of the VMI drawings away from the standard scoring and assessment, based on static measures, and uses dynamic properties of the drawing to elicit underlying movement differences between Dyspraxic children and control subjects. Rapid-Aimed movements are introduced as a hypothesis on the production of movement. Software has been developed to model the velocity curves of children's drawings based on this hypothesis. We have shown that Rapid-Aimed movement analysis is not practical or useful for data gathered in this fashion.

Chapter 7 continues from Chapter 6 in the analysis of Dynamic information. Extraction of alternative drawing features not available without computer intervention is further investigated. The exploitation of these features as indicators of the development of visuo-motor skills is discussed. Software has been developed to extract features from the velocity profile of pen movements. We have assessed the drawing strategy for a pair of shapes which share similar component lines. We show that the maximum velocity reached during the movement task, and the standard deviation of velocity, are the best features for discriminating visuo-motor difficulties.

Overall, this thesis has presented several approaches which could be applied to assist the diagnosis of Dyspraxia. It has been demonstrated that recording a child's drawings in real time is a useful addition to existing tests. This research indicates that the efficiency and effectiveness of tests, such as the Beery Developmental Test of Visual Motor Integration, may be increased by capturing data in this way.

## **8.2 Suggestions for Future Research**

Finally, some suggestions for consideration as possible areas of further research are presented below.

1. The VMI test can be administered to several children at the same time, for example in a school class. Using a digitising tablet would require a computer for each child. A useful extension to the Assisted Scorer software would be to accept scanned images of the test drawings. This would allow ordinary tests, which have been carried out using the standard VMI procedure, to be assessed using this software.

After scanning the completed VMI test booklet, it would be necessary to identify the positions of the drawings and identify which shape was copied in each drawing. It would be possible using simple image processing techniques to identify the six boxes on each page of the booklet. The template shapes are provided on the page and these could be classified with a very high degree of accuracy. It is therefore feasible to have a system which would take a scanned image and be able to extract and label the subjects drawings automatically.

2. The General Scoring software returns a pass/fail decision for each shape assessed. A logical progression of this would be to produce a pass/fail decisions for each of the criteria specified. This could be achieved by dividing the absolute rule commands into groups. Each group assessing a different criterion.
3. Instead of using the simple test harness to administer the scoring, the scorer software component could be integrated into the Assisted Scoring software described in Chapter 3. The user friendly environment provided by this software could be combined with an automated scoring facility.
4. The scorer software could be modified to create a semi-automated system. The initial feature detection section of the software could be augmented, or even

replaced, by allowing a user to accurately pick out the cluster positions. This would increase the accuracy of the scoring, but would sacrifice some of the benefits of a fully automated system.

5. The absolute scoring rules could be implemented in a Dynamic Linked Library (DLL). This would enable rule sets to be implemented without recompiling the rest of the scoring program. Extra rules could then be added by third parties who didn't have access to the entire program source code. Different libraries could then be loaded for scoring different projects, easing the trouble of duplicate rule names. This would allow the scorer to be more easily adapted for different assessment tests.
6. As shown in Chapter 6, the VMI test does not produce movements which are suitable for analysis using rapid-aimed movement modelling. The parameter estimation software, however, could be applied to a more specifically designed test. Meyer et al. (1998) describe an experimental procedure for capturing rapid-aimed movements from pen actions which isolate wrist rotation.

### **8.3 Summary**

This work has shown that software tools can be effectively incorporated into an established figure copying test. The benefits of such tools have been discussed and recommendations made for future research. The use of novel features not discernable in the standard test has also been discussed.

It is hoped that further research into developmental motor difficulties will be able to use the software, and methods developed during this work, to further the insight into the nature of this disorder.

# Bibliography

Adam, J., Nieuwenstein, R., Paas, F., Kingma, H., Willems, P., Werry, M. (2000). *Journal of Experimental Psychology: Human Perception and Performance*. 26, 1, 295-312.

Anastasi, A. (1968). *Psychological Testing*. Macmillan, 3<sup>rd</sup> Edition.

Ayres, A. (1985). *Development of apraxia and adult onset apraxia*. Sensory Integration International, Torrence, California, USA.

Ayyash, H., Preece, P. (2003). Evidence-based treatment of motor co-ordination disorder. *Current Paediatrics*, 13, 360-364.

Beaumont, G. (1982). System requirements for interactive testing. *International Journal of Man-Machine Studies*, 17, 311-320.

Beery, K. (1997). *Developmental Test of Visual-Motor Integration: Administration scoring and teaching manual* (4<sup>th</sup> Edition). Modern Curriculum Press, Cleveland, Ohio, USA.

Brault, J., Plamondon, R. (1993). Segmenting Handwritten Signatures at Their Perceptually Important Points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 15, 9, 953-957.

- Brucq, D., Amara, M., Ruiz, V. (1997). On-line handwritten segmentation in linear drawings. *Pattern Recognition Letters*, 18, 193-202.
- Bruininiks, R. (1978). *Bruininiks-Oseretsky Test of Motor Proficiency*. American Guidance Services, Circle Pines, Minnesota, USA.
- Cantell, M., Smyth, M., Ahonen, T. (2003). Two distinct pathways for developmental coordination disorder: Persistence and resolution. *Human Movement Science*, 22, 413-431.
- Cermak, S. (1985). Developmental Dyspraxia. *Neuropsychological studies of apraxia and related disorders*. 225-248.
- Cermak, S., Costers, W., Drake, C. (1980). Representational and non-representational gestures in boys with learning disabilities. *American Journal of Occupational Therapy*, 34, 19-26.
- Chen, T., Chung, K. (2001). An Efficient Randomized Algorithm for Detecting Circles. *Computer Vision and Image Understanding*, 83, 172-191.
- Chiu, S., Liaw, J. (2005). An effective voting method for circle detection. *Pattern Recognition Letters*, 26, 2, 121-133
- Coleman, R., Piek, J., Livesey, D. (2001). A longitudinal study of motor ability and kinaesthetic acuity in young children at risk of developmental coordination disorder. *Human Movement Science*, 20, 95-110.
- Conrad, K., Cermak, S., Drake, C. (1983). Differentiation of praxis among children. *American Journal of Occupational Therapy*, 37, 466-473.
- Cousins, M., Smyth, M. (2003). Developmental coordination impairments in adulthood. *Human Movement Science*, 22, 433-459.

DCM-IV (1994) *Diagnostic and Statistical Manual of Mental Disorders* (4<sup>th</sup> Edition). American Psychiatric Association, Washington DC, USA.

Denckla, M., Roeltgen, D. (1992). Disorders of motor function and control. *Handbook of neuropsychology*, 6, 455-476.

Dewey, D. (1995). What is Developmental Dyspraxia? *Brain and Cognition*, 29, 254-274.

Dewey, D., Kaplan, B. (1994). Subtyping of developmental motor deficits. *Developmental Neuropsychology*, 10, 265-284.

Dewey, D., Kaplan, B. (1994). Subtyping of developmental motor deficits. *Journal of Clinical and Experimental Neuropsychology*, 10, 265-284.

Dhanish, P. (2002). A simple algorithm for evaluation of minimum zone circularity error from coordinate data. *International Journal of Machine Tools and Manufacture*, 42, 1589-1594.

Dijoua, M., Plamondon, R. (2005). Delta-lognormal parameter estimation by non-linear regression and evolutionary algorithm: A comparative study. *Advances in Graphonomics: Proceedings of the International Graphonomics Society*, 12, 44-48.

Djeziri, S., Guerfali, W., Plamondon, R., Robert, J. (2000). Learning handwriting with pen-based systems: computational issues. *Pattern Recognition*, 35, 1049-1057.

Gardiner, M. (1989). *Test of Visual-Perceptual Skills*. Health Publishing Company, San Francisco, California, USA.

Geuze, R., Borger, H. (1993). Children who are clumsy: Five years later. *Adapted Physical Activity Quarterly*, 10, 10-21.



Gillberg, C. (1998). Hyperactivity, inattention and motor control problems: Prevalence, comorbidity and background factors. *Folia Phoniatrica et Logopedia*, 50, 107-117.

Gillberg, C., Kadesjo, B. (1998). AD/HD and developmental coordination disorder. *Attention deficit disorders and comorbidities in children, adolescents and adults*. American Psychiatric Press, Washington DC, USA.

Gubbay, S. (1975). *The Clumsy Child*. W.B. Saunders, New York, USA.

Hamilton, S. (2002). Evaluation of Clumsiness in Children. *American Family Physician*, 66, 8, 1435-1440

Hay, J., Hawes, R., Faight, B. (2004). Evaluation of a Screening Instrument for Developmental Coordination Disorder. *Journal of Adolescent Health*, 34, 308-313

Henderson, L., Rose, P., Henderson, S. (1992). Reaction time and movement time in children with developmental coordination disorder. *Journal of Child Psychology and Psychiatry*, 33, 895-905.

Henderson, S., Sugden, D. (1992). *Movement Assessment Battery for Children*. The Psychological Corporation, Sidcup, UK.

Hoare, D. (1994). Subtypes of developmental coordination disorder [Special issue: Developmental coordination disorder]. *Adapted Physical Activity Quarterly*, 11, 158-169.

Hollerbach, J. (1981). An Oscillation Theory of Handwriting. *Biological Cybernetics*, 39, 139-156.

Ioannou, D., Huda, W., Laine, A. (1999). Circle recognition through a 2D Hough Transform and radius histogramming. *Image and Vision Computing*, 17, 15-26.

Jywe, W., Liu, C., Chen, C. (1999). The min-max problem for evaluating the form error of a circle. *Measurement*, 26, 273-282.

Kadesjo, B., Gillberg, C. (1999). Developmental coordination disorder in Swedish 7-year-old children. *Journal of the American Academy of Child & Adolescent Psychiatry*, 38, 820-828.

Kaplan, B., Crawford, S., Wilson, B., Dewey, D. (1997). Comorbidity of developmental coordination disorder and different types of reading disability. *Journal of the International Neuropsychological Society*, 3, 54.

Kaplan, B., Wilson, B., Dewey, D., Crawford, S. (1998). DCD may not be a discrete disorder. *Human Movement Science*, 17, 471-490.

Kim, N., Kim, S. (1995). Geometrical Tolerances: Improved linear approximation of least squares evaluation of circularity by minimum variance. *International Journal of Machine Tools Manufacture*, 36, 3, 355-366.

Lelivelt, A., Meulenbroek, R., Thomassen, A. (1996). Mapping Abstract Main Axes in Handwriting to Hand and Finger Joints. *Handwriting and Drawing Research: Basic and Applied Issues*, 29-40.

MacKenzie, C., Marteniuk, R., Dugas, C., Liske, D., Eickmeier, B. (1987). Three-dimensional Movement Trajectories in Fitts' task: Implications for control. *The Quarterly Journal of Experimental Psychology*, 39A, 629-647.

Macnab, J., Miller, L., Polatajko, H. (2001). The search for subtypes of DCD: Is cluster analysis the answer? *Human Movement Science*, 20, 49-72.

Mandich, A., Buckolz, E., Polatajko, H. (2002). On the ability of children with developmental coordination disorder (DCD) to inhibit response initiation: The simon effect. *Brain and Cognition*, 50, 150-162.

Mandich, A., Polatajko, H. (2003). Developmental coordination disorder: Mechanisms, measurement and management. *Human Movement Science*, 22, 407-411.

- Marquardt, D. (1963). An algorithm for least squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11, 431-441.
- Maruff, P., Wilson, P., Trebilcock, M., Currie, J. (1999). Abnormalities of imagined motor sequences in children with developmental coordination disorder. *Neuropsychologia*, 37, 1317-1324.
- Meulenbroek, R., Thomassen, A. (1991). Stroke-direction preferences in drawing and handwriting. *Human Movement Science*, 10, 247-270.
- Meyer, D., Kornblum, S., Abrams, R., Wright, C., Smith, J. (1988). Optimality in Human Motor Performance: Ideal Control of Rapid Aimed Movements. *Psychological Review*, 95, 3, 340-370.
- Miller, L., Missiuna, C., Macnab, J. (2001). Clinical description of children with developmental coordination disorder. *Canadian Journal of Occupational Therapy*, 68, 5-15.
- Miyahara, M. (1994). Subtypes of students with learning disabilities based upon gross motor functions. *Adapted Physical Quarterly*, 11, 170-178.
- Mon-Williams, M., Wann, J., Pascal, E. (1999). Visual-proprioceptive mapping in children with developmental coordination disorder. *Developmental Medicine & Child Neurology*, 41, 247-254.
- Novaski, O., Barczak, A. (1997). Utilization of Voronoi diagrams for circularity algorithms. *Precision Engineering*, 20, 188-195.
- Piek, J., Murray, J., Nieman, A., Anderson, M., Hay, D., Smith, L., McCoy, M., Hallmayer, J. (2004). The relationship between motor coordination, executive functioning and attention in school aged children. *Archives of Clinical Neuropsychology*, 19, 8, 1063-1076.

Pilu, M., Fitzgibbon, A., Fisher, R. (1996). Ellipse-specific direct least-square fitting. *IEEE International Conference on Image Processing*.

Plamondon, R. (1991). On the Automatic Extraction of Biomechanical Information from Handwriting Signals. *IEEE Transactions on Systems, Man, and Cybernetics*, 21, 1, 90-101.

Plamondon, R. (1995a). A kinematic theory of rapid human movements: Part I. Movement representation and generation. *Biological Cybernetics*, 72, 295-307.

Plamondon, R. (1995b). A kinematic theory of rapid human movements: Part II. Movement time and control. *Biological Cybernetics*, 72, 309-320.

Plamondon, R. (1997). A kinematic theory of rapid human movements: Part III. Kinetic Outcomes. *Biological Cybernetics*, 78, 133-145.

Plamondon, R., Privitera, C. (1995). A neural model for generating and learning a rapid movement sequence. *Biological Cybernetics*, 74, 117-130.

Qiao, Y., Ong, S. (2004). Connectivity-based multiple-circle fitting. *Pattern Recognition*, 37, 755-765.

Rémi, C., Frélicot, C., Courtellemont, P. (2002). Automatic analysis of the structuring of children's drawings and writing. *Pattern Recognition*, 35, 1059-1069.

Rodger, S., Ziviani, J., Watter, A., Woodyatt, G., Springfield, E. (2003). Motor and functional skills of children with developmental coordination disorder: A pilot investigation of measurement issues. *Human Movement Science*, 22, 461-478.

Samuel, G., Shunmugam, M. (2000). Evaluation of circularity from coordinate and form data using computational geometric techniques. *Precision Engineering*, 24, 251-263.

- Semel, E., Wiig, E., Second, W. (1987). *The Clinical Evaluation of Language Fundamentals*. The Psychological Corporation, New York, USA.
- Shimaya, A. (1997). Perception of Complex Line Drawings. *Journal of Experimental Psychology: Human Perception and Performance*, 23, 1, 25-50.
- Shoemaker, M., Kalverboer, A. (1994). Social and affective problems of children who are clumsy: How early do they begin? *Adapted Physical Activity Quarterly*, 11, 130-140.
- Smith, R. (1987). Computer Processing of line images: A Survey. *Pattern Recognition*, 20, 1, 7-15.
- Smits-Engelsman, B., Niemeijer, G., van Galen, G. (2001). Fine motor deficiencies in children diagnosed as DCD based on poor grapho-motor ability. *Human Movement Science*, 20, 161-182.
- Smits-Engelsman, B., Wilson, P., Westenberg, Y., Duysens, J. (2003). Fine motor deficiencies in children with developmental coordination disorder and learning disabilities: An underlying open-loop control deficit. *Human Movement Science*, 22, 495-513.
- Swanson, K., Lee, D., Wu, V. (1995). An optimal algorithm for roundness determination on convex polygons. *Computational Geometry*, 5, 225-235.
- Thompson, J., Wilson, S. (1982). Automated psychological testing. *International Journal of Man-Machine Studies*, 17, 279-289.
- Van Gemmert, A., Van Galen, G. (1997). Stress, Neuromotor Noise, and Human Performance: A Theoretical Perspective. *Journal of Experimental Psychology: Human Perception and Performance*, 23, 5, 1299-1313.
- Visser, J. (2003). Developmental coordination disorder: a review of research on subtypes and comorbidities. *Human Movement Science*, 22, 479-493.

Volans, P. (1982). Pros and cons of tailored testing: an examination of issues highlighted by experience with an automated testing system. *International Journal of Man-Machine Studies*, 17, 301-304.

Wang, M., Cheraghi, H., Masud, A. (1999) Circularity error evaluation theory and algorithm. *Precision Engineering*, 23, 164-176.

Wechsler, D. (1991). *Wechsler Intelligence Scale for Children*, (3<sup>rd</sup> Edition). The Psychological Corporation, New York, USA.

Zhu, L., Ding, H., Xiong, Y. (2003). A steepest descent algorithm for circularity evaluation. *Computer-Aided Design*, 35, 255-265.

