


2017

Data Mining by Grid Computing in the Search for Extrasolar Planets

Oisín Creaner [Thesis]

Dublin Institute for Advanced Studies, creanero@cp.dias.ie

Follow this and additional works at: <https://arrow.tudublin.ie/ittthedoc>

 Part of the [Databases and Information Systems Commons](#), [External Galaxies Commons](#), [Instrumentation Commons](#), [Numerical Analysis and Scientific Computing Commons](#), [Other Astrophysics and Astronomy Commons](#), [Other Computer Sciences Commons](#), [Software Engineering Commons](#), [Stars, Interstellar Medium and the Galaxy Commons](#), and the [Theory and Algorithms Commons](#)

Recommended Citation

Creaner, O. (2017) Data Mining by Grid Computing in the Search for Extrasolar Planets, Doctoral Thesis, Institute of Technology, Tallaght, Dublin. DOI:10.21427/7w45-6018

This Theses, Ph.D is brought to you for free and open access by the Theses&Dissertations at ARROW@TU Dublin. It has been accepted for inclusion in Doctoral by an authorized administrator of ARROW@TU Dublin. For more information, please contact yvonne.desmond@tudublin.ie, arrow.admin@tudublin.ie, brian.widdis@tudublin.ie.



This work is licensed under a [Creative Commons Attribution-NonCommercial-Share Alike 3.0 License](#)

Data Mining by Grid Computing in the Search for Extrasolar Planets



Oisín Oliver Creaner B.A. (mod)

Being a Thesis presented for the award of Doctor of Philosophy

Dr. Eugene Hickey CPhys, MInstP, DEA, PhD

Kevin Nolan BSc MSc, Dr. Niall Smith Ph.D

School of Science and Computing
Institute of Technology, Tallaght, Dublin
Old Belgard Road,
Tallaght,
Dublin 24

Submitted to Quality and Qualifications Ireland (QQI) July 2016

i. Declaration Statement

I hereby certify that the material, which I now submit for assessment on the programmes of study leading to the award of PhD, is entirely my own work and has not been taken from the work of others except to the extent that such work has been cited and acknowledged within the text of my own work. No portion of the work contained in this Thesis has been submitted in support of an application for another degree or qualification to this or any other institution.

Signature of Candidate

Date

I hereby certify that all the unreferenced work described in this Thesis and submitted for the award of PhD, is entirely the work of Oisín Creaner. No portion of the work contained in this Thesis has been submitted in support of an application for another degree or qualification to this or any other institution.

Signature of Supervisor

Date

Signature of Supervisor

Date

Signature of Supervisor

Date

ii. Acknowledgements

I would like to take this opportunity to thank everyone who has helped me through these last few years, and without whom this Thesis would never have made it to publication.

First, I would like to thank my supervisors, Dr. Eugene Hickey and Kevin Nolan – Eugene, your support and encouragement throughout have kept me going and Kevin, without your constant drive towards perfection and robust work has made this project what it is. I would like to thank everyone at the Institute of Technology, Tallaght, especially John Behan, Martin McCarrick and Tadhg O Briain, for the opportunity to complete this project, in particular through the provision of the PhD continuation fund.

Second, I thank the entire team down in Blackrock Castle Observatory, most of all Dr. Niall Smith, my co-supervisor, for all the work which you have assisted us with throughout the years.

Thanks are also due to the team at Grid Ireland in Trinity College Dublin, especially John Smith, Stephen Childs, John Ryan, David O’Callaghan and Gabriele Pierantoni for their help in bringing an astrophysicist into the world of High Performance Computing.

This project made extensive use of data from SDSS. Funding for the SDSS and SDSS-II has been provided by the Alfred P. Sloan Foundation, the Participating Institutions, the National Science Foundation, the U.S. Department of Energy, the National Aeronautics and Space Administration, the Japanese Monbukagakusho, the Max Planck Society, and the Higher Education Funding Council for England. The SDSS Web Site is <http://www.sdss.org/>.

I thank my family for being there for me throughout it all, my parents Ann and Aidan, who first encouraged me to look to the stars, my brothers Dónal and Rúairí who kept me sane throughout the long years of work, my grandmother Miriam who provided me with a refuge away from the city in which to work, my nephew and many aunts, uncles, cousins and friends who have been there for me when needed.

Finally, I dedicate this work to two people who were there for me from the start but who weren’t able to see me finish this work. To my granddad Pat and my nanny Peg, I hope that you’re proud of me.

iii. Abbreviations List

Abbreviation	Meaning
ΔC	Difference in colour
ΔC_{\max}	Maximum permitted difference in colour
Δm_{\max}	Maximum permitted difference in magnitude
Δm	Difference in magnitude
AD	Anno Domini
ADASS	Astronomical Data Analysis Software and Systems
API	Applications Programming Interface
ASCII	American Standard Code for Information Interchange
AT&T	American Telephone & Telegraph
B	Johnson B magnitude
Bash	Bourne-Again SHell
BC	Before Christ
BCO	Blackrock Castle Observatory
CAS	Catalogue Access Server
CCD	Charge Coupled Device
cf.	<i>conferre</i> (compare with)
char	Character
CIT	Cork Institute of Technology
C-K	Comparison-Check Photometry
C_l	Colour with respect to the next band of longer wavelength
CLI	Command Line Interpreter/Interface
C_s	Colour with respect to the next band of shorter wavelength
CSV /.csv	Comma Separated Value
CTI	Catalogue Information File
DAS	Data Access Server
DE	Development Environment
Dec	Declination
double	Double Precision Floating Point Number
DR	Data Release
e.g.	<i>exempli gratia</i> (for example)
EC2	Elastic Cloud Compute
EGI	European Grid Infrastructure
etc.	<i>et cetera</i> (and so on)
FITS / .fits /.fit	Flexible Image Transfer System
FORTTRAN	FORmula TRANslation (programming language)

FoV	Field of view (telescope specific)
g	SDSS g band magnitude
GB	GigaByte (10^9 or 2^{30} Bytes)
GMS	Grid Management System
GUI	Graphical User Interface
GUID	Globally Unique IDentifier
HDD	Hard Disk Drive
HDU	Header Data Unit
HPC	High Performance Computing
i	SDSS i band magnitude
IDE	Integrated Development Environment (IDE)
int	Integer
ITTD	Institute of Technology, Tallaght, Dublin
JDL	Job Description Language
JSS	Job Submission System
kB	kiloByte (10^3 or 2^{10} Bytes)
ksh	Korn SHell
LAN	Local Area Network
LFC	Logical File Catalogue
lfc-cp	gLite command. Analogous to cp
lfc-cr	gLite command. Copy and Register.
lfc-ls	gLite command. Analogous to ls
LFN	Logical File Name
long	Long form integer
ls	Unix "list" tool. Lists the contents of a directory
LSST	Large Synoptic Sky Survey
max	maximum
MB	MegaByte (10^6 or 2^{20} Bytes)
min	minimum
m_n	magnitude in a given band (n)
m_{n+1}	magnitude in the next band of longer wavelength than a given band (n)
m_{n-1}	magnitude in the next band of shorter wavelength than a given band (n)
MPI	Message Passing Interface
MS	MicroSoft
N	Number: usually an integer which refers to a counted quantity
PC	Personal Computer
Pol	Point of Interception
PPR/.ppr	Pipeline Parameter File

PRM/.prm	API Parameter File
PSF	Point-Spread Function
QE	Quantum Efficiency
r	SDSS r band magnitude
R	Rating
RA	Right Ascension
RAM	Random Access Memory
R _l	Rating with respect to the next band of longer wavelength
R _s	Rating with respect to the next band of shorter wavelength
S	Size (of FoV)
S3	Simple Storage Service
SCG	Scientific Computing Group
SDSS	Sloan Digital Sky Survey
sh	bourne SHell
SNR	Signal-to-Noise Ratio
SQL	Structured Query Language
SSH	Secure SHell
SURL	Storage Universal Resource Locator
TB	TeraByte (10^{12} or 2^{40} Bytes)
TCD	Trinity College, Dublin
tsObj	SDSS calibrated object lists
TTV	Transit Time Variation
TXT/.txt	Text file
u	SDSS u band magnitude
U	Johnson U magnitude
UFS	Unix File System
USNO	United States Naval Observatory
V	Johnson V magnitude
VLDB	Very Large Data Base
VM	Virtual Machine
VO	Virtual Organisation
WAN	Wide Area Network
wc	Unix word count tool
WN	Worker Node
XLDB	eXtremely Large Data Base
z	SDSS z band magnitude

iv. Abstract

A system is presented here to provide improved precision in ensemble differential photometry. This is achieved by using the power of grid computing to analyse astronomical catalogues. This produces new catalogues of optimised pointings for each star, which maximise the number and quality of reference stars available.

Astronomical phenomena such as exoplanet transits and small-scale structure within quasars may be observed by means of millimagnitude photometric variability on the timescale of minutes to hours. Because of atmospheric distortion, ground-based observations of these phenomena require the use of differential photometry whereby the target is compared with one or more reference stars. CCD cameras enable the use of many reference stars in an ensemble. The more closely the reference stars in this ensemble resemble the target, the greater the precision of the photometry that can be achieved.

The Locus Algorithm has been developed to identify the optimum pointing for a target and provide that pointing with a *score* relating to the degree of similarity between target and the reference stars. It does so by identifying potential points of aim for a particular telescope such that a given target and a varying set of references were included in a field of view centred on those pointings. A score is calculated for each such pointing. For each target, the pointing with the highest score is designated the optimum pointing.

The application of this system to the Sloan Digital Sky Survey (SDSS) catalogue demanded the use of a High Performance Computing (HPC) solution through Grid Ireland. Pointings have thus been generated for 61,662,376 stars and 23,697 quasars.

v. Table of Contents

i. Declaration Statement	ii
ii. Acknowledgements.....	iii
iii. Abbreviations List.....	iv
iv. Abstract	vii
v. Table of Contents	viii
vi. Table of Tables	xv
vii. Table of Figures.....	xviii
1. Introduction	1
1.1. Introduction to the Project.....	1
1.2. Structure of the Thesis	2
1.2.1. Part I: Background	3
1.2.2. Part II: Design & Implementation.....	5
1.2.3. Part III: Results	7
1.2.4. Part IV: Future Work & Conclusions	8
1.2.5. Appendices	9
1.3. Summary	10
2. Core Concepts of Photometric Variability	11
2.1. Variability of Astronomical Objects	11
2.1.1. Variable Stars	13
2.1.2. Exoplanets	17
2.1.3. Quasars	24
2.2. Photometry	25
2.2.1. Classes of Photometry.....	25
2.2.2. Historical Development of Photometry	27
2.2.3. Differential Photometry Operations.....	30
2.3. Application of Data Mining Techniques to Astronomical Catalogues	38
2.4. Conclusions	39
3. Computing Background & Concepts	40
3.1. Assessment of Computing Requirements	41
3.2. High Performance Computing	41
3.2.1. Grid Ireland	44

3.3.	Software Specifications.....	47
3.3.1.	Computer Programming Concepts.....	47
3.3.2.	Software Solutions	52
3.4.	Conclusion	60
4.	Project Objectives	61
4.1.	Astronomical Goals.....	61
4.1.1.	Individual Field Optimisation	62
4.1.2.	Production of Catalogue of Optimised Fields.....	63
4.1.3.	Meta-Analysis	66
4.2.	Computing Goals	68
4.2.1.	Demonstration of Grid Use on Data-Rich, Process Poor Problem	68
4.2.2.	Computing Metrics	69
4.3.	Summary	71
5.	Locus Algorithm & Scoring	73
5.1.	Coordinate System	74
5.2.	Locus Algorithm Definition.....	77
5.2.1.	Aggregate Data from SDSS	77
5.2.2.	Identify Potential Reference Stars.....	78
5.2.3.	Apply the Locus to Each Candidate.....	79
5.2.4.	Identify the Points of Intersection.....	81
5.2.5.	Output the Intersection with the Best Score.....	82
5.2.6.	Failed Targets.....	83
5.3.	Scoring System	84
5.3.1.	Rating System Options.....	84
5.3.2.	Combining Ratings into Scores.....	88
5.4.	Conclusion	89
6.	Project Design Concepts and Approach	90
6.1.	Design Concepts	90
6.1.1.	Design Strategy	91
6.1.2.	Design Philosophy	93
6.2.	Design Approach.....	95
6.2.1.	Design Process	96

6.2.2.	Design Techniques	98
6.3.	Summary	100
7.	Project Design.....	102
7.1.	Design Requirements & Constraints.....	102
7.1.1.	Astronomical Specifications	102
7.1.2.	Computational Limitations	104
7.1.3.	Consequent Software Design	107
7.2.	Overall Software Design	109
7.2.1.	SDSS Data Access API.....	111
7.2.2.	Data Pipeline	116
7.2.3.	Parameterisation	125
7.2.4.	Grid Management Software	134
7.2.5.	SQL Queries to CAS.....	136
7.2.6.	Error Handling Routines	138
7.3.	Summary	140
8.	Data Storage and Management	143
8.1.	Data Storage Devices	143
8.1.1.	Logical File Catalogue	145
8.1.2.	Grid Data Transfers.....	146
8.2.	Directory Structure.....	147
8.2.1.	Top Level Structure.....	147
8.2.2.	Workspace.....	148
8.2.3.	Scripts.....	149
8.2.4.	Test.....	150
8.2.5.	Data	151
8.2.6.	Release	154
8.3.	Internal Structure of File Types	155
8.3.1.	Flexible Image Transport System (FITS)	156
8.3.2.	Catalogue Information (CTI)	161
8.3.3.	Parameter Files.....	161
8.3.4.	Text (TXT).....	164
8.3.5.	Comma Separated Value (CSV)	165

8.3.6.	Job Description Language (JDL)	168
8.4.	Summary	169
9.	Implementation & Operations	172
9.1.	Project Implementation	172
9.1.1.	Development Environment	172
9.1.2.	Software Testing	173
9.2.	Practicalities of Grid Operation	174
9.2.1.	Grid Job Submission & Scripts	175
9.2.2.	Data Limitations of the Grid	176
9.2.3.	Result Generation & Collation.....	178
9.3.	Summary	179
10.	Individual Result	181
10.1.	Aggregate Data from SDSS	182
10.2.	Identify Potential Reference Stars.....	183
10.3.	Apply the Locus Algorithm to each Candidate.....	187
10.4.	Observation of a Pointing.....	190
10.5.	Expansion to the Catalogue on the Grid	191
10.6.	Conclusions	192
11.	Catalogue Outputs & Analysis.....	194
11.1.	Overview of the Catalogues Generated.....	194
11.2.	Local Catalogue	196
11.3.	Quasar Catalogue	198
11.4.	Exoplanet Catalogue	201
11.5.	Summary	204
12.	Meta-Analysis of the Exoplanet Catalogue.....	205
12.1.	Magnitude Variations.....	206
12.1.1.	Variation of Score with Magnitude.....	208
12.1.2.	Variation of Number of Failed Targets with Magnitude	209
12.1.3.	Variation of Descriptive Statistics	210
12.2.	Colour Variation.....	212
12.2.1.	Variation of Score with Colour.....	213
12.2.2.	Variation of Number of Failed Targets with Colour.....	215

12.2.3. Variation of Descriptive Statistics with Colour	216
12.3. Summary	217
13. Computational Results	221
13.1. Data Metrics	221
13.1.1. Source Data Metrics	223
13.1.2. API Data Metrics.....	223
13.1.3. Pipeline Data Metrics	225
13.2. Processing Metrics	227
13.2.1. API Metrics	228
13.2.2. Pipeline Metrics	229
13.3. Grid Metrics	233
13.3.1. Local Catalogue Job	235
13.3.2. Quasar Catalogue Job.....	236
13.3.3. Exoplanet Catalogue Job.....	237
13.4. Issues Arising from Project.....	239
13.5. Conclusions	242
14. Future Use & Refinements of the Project	244
14.1. Reuse of the Project	245
14.1.1. Magnitude and Colour Arguments.....	246
14.1.2. Choice of SDSS filter.....	248
14.1.3. Observational Parameters.....	248
14.1.4. Target List Options	251
14.1.5. Data Source Parameters	252
14.2. Migration to Cloud.....	252
14.3. Refinement of the Project	256
14.3.1. Addition of Functions to the Algorithm.....	257
14.3.2. Scoring	265
14.3.3. Computational Optimisation	268
14.4. Conclusions	271
15. Conclusions	273
Appendix A Bibliography & References	276
A-a In-Text citations	276

A-b	SDSS Acknowledgement	299
Appendix B	Symbols and Terminology	301
B-a	Hierarchy of the Thesis	301
B-b	Text Conventions	301
B-c	Design Symbols	302
B-c-a	Programs and Scripts	302
B-c-b	Functions	303
B-c-c	Processes	303
B-c-d	Loops.....	303
B-c-e	Logical Operations	304
B-c-f	Computing Elements.....	304
B-c-g	Storage Devices.....	304
B-c-h	Stored Data.....	305
B-c-i	Logical Catalogues.....	305
B-c-j	Data Flow	305
B-c-k	Logical Operations	306
B-c-l	Developmental Tools	306
B-c-m	Documents	306
Appendix C	Code Sample	cccvii
C-a	C.....	cccvii
C-b	BASH	cccviii
C-c	JDL.....	cccxcv
C-d	SQL	cccxcvi
Appendix D	Results Samples	cccxi
D-a	Distribution of SDSS flags.....	cccxi
D-b	Local Catalogue Data.....	cccxi
D-c	Output Catalogues.....	cccxi
D-c-a	Sample of Quasar Catalogue.....	cccxi
D-c-b	Sample of Exoplanet Catalogue.....	cccxi
D-c-c	Reference Stars for SDSS J203733.62+001953.5	cccxi
Appendix E	Index.....	cccxi

vi. Table of Tables

Table 2-1: Breakdown of number and proportion of planets detected by various techniques. Taken from the Interactive Extra-solar Planets Catalog, Zolotukin, 2017. Retrieved 25 th January 2017 [9]	17
Table 3-1: Comparison between Grid and Cluster solutions. Green indicates the superior option in a category, while red indicates the inferior.....	43
Table 3-2: Comparison of SDSS Data Access Options. Green highlighting indicates preferred option.....	54
Table 3-3: Comparison of programming languages with FITSIO Capability	56
Table 7-1: Reduced version of SDSS Clean Sample of Point Sources Filter [143]	116
Table 7-2: Error Definitions. Errors defined in this project use negative integers. FITSIO Built-in errors use positive integers to report errors	139
Table 8-1: Illustration of the encoding of exoplanet job parameters in an output file name. Colours are used to indicate the data components to which each component of the filename corresponds.....	152
Table 8-2: FITS Data Types [118]	157
Table 10-1: Table of reference stars for SDSS J113824.40+483457.8 (highlighted in yellow.) One of the reference stars, SDSS J113749.38+482307.1, is shown in red and is used to provide a worked example of the rating and scoring system.....	190
Table 11-1: Excerpt from the Local Catalogue. Columns are RA, Dec and Magnitude (u, g, r, i, z).....	196
Table 11-2: Comparison between Source Catalogue, Local Catalogue, and output from the SDSS Clean Sample of Stars Algorithm based on SQL query to the CAS [6].....	197
Table 11-3: Parameter list for Quasar Catalogue.....	198
Table 11-4: Excerpt from Quasar Catalogue. Columns are Right Ascension, Declination, Magnitude (u,g,r,i,z), Pointing RA, Pointing Dec and Score. Highlighted	

in red are two quasars for which no suitable pointings were possible for the given criteria. Highlighted in green is the quasar with the best score in this small sample, SDSS J170355.79+604511.7	198
Table 11-5: Summary of output from Quasar Catalogue.....	199
Table 11-6: Descriptive statistics of the Quasar Catalogue, filtered to those quasars for which pointings were available. The only magnitude (g, r, i) and colour parameters (g-r, r-i) which contribute to the score are shown.....	200
Table 11-7: Summary of Input Parameters for Exoplanet Catalogue.....	201
Table 11-8: Summary of output from Exoplanet Catalogue. The percentage column compares elements in the output catalogue with the number of unique objects in the overall catalogue and with the number of objects in the local catalogue.....	201
Table 11-9: Excerpt from Exoplanet Catalogue. Columns are Right Ascension, Declination, Magnitude (u,g,r,i,z), Pointing RA, Dec and Score. Highlighted in green is the highest scoring target in this sample	203
Table 11-10: Descriptive statistics of a subset of 10^6 stars of the Exoplanet Catalogue, filtered to include only those 942,895 stars for which pointings were available. Only the magnitude (g, r, i) and colour parameters (g-r, r-i) which contribute to the score are shown.	203
Table 13-1: Data Size for the various databases used and created during this project. Approximate or rounded values are indicated with a tilde (~), while values with significant trailing zeroes are indicated with a decimal point (.). The Mean Size per entry is calculated by dividing the total size of the catalogue by the number of entries and as such includes contributions from header data spread per entry.....	222
Table 13-2: Assessment of processing elements of three phases of the project	228
Table 13-3: Grid metrics for the three primary grid jobs. Timing data for the Quasar job is unavailable at this time.....	234
Table 14-1: Summary of proposed expansions to this project.....	271

Table 15-1: Distribution of <code>flags</code> bit in 62630 SDSS fits entries.....	cccxi
Table 15-2: Distribution of <code>flags_2</code> bit in 62630 SDSS fits entries	cccxx
Table 15-3: Distribution of <code>status</code> bit in 62630 SDSS fits entries	cccxx
Table 15-4: An arbitrary sample of 50 entries in the Local Catalogue.....	cccxi
Table 15-5: The top 40 quasars in the Quasar Catalogue	cccxi
Table 15-6: the top 50 stars by score in the sample of 10^6 targets from the Exoplanet Catalogue.....	cccxi
Table 15-7: Coordinate (RA, Dec) list for all 247 references for SDSS J203733.62+001953.5, the target with the highest score in the sample of 10^6 targets from the Exoplanet Catalogue.....	cccxi

vii. Table of Figures

Figure 1-1: Thesis Structure.....	3
Figure 1-2: Background Chapters	4
Figure 1-3: Design and Implementation Chapters	5
Figure 1-4: Results Chapters.....	7
Figure 1-5: Future Work Chapters	9
Figure 1-6: Appendices	9
Figure 2-1: Calculated duration, shape and depth of light curves for uniformly dark, centrally eclipsing secondary objects of various radii orbiting uniformly illuminated primary objects at constant velocity.....	14
Figure 2-2: Calculated duration, shape and depth of light curves for uniformly dark secondary objects of constant radii ($r_{\text{secondary}} = 0.5R_{\text{primary}}$) transiting the primary at different apparent latitudes.....	16
Figure 2-3: Exoplanet Detections by method as a function of time. Taken from the Interactive Extra-solar Planets Catalog, Zolotukin, 2013-2017. Retrieved 25 th January 2017 [9]	19
Figure 2-4: (left) An image of Venus in transit across the sun, demonstrating a planetary transit and limb darkening in the Sun. Image:NASA/SDO, HMI [34].....	20
Figure 2-5: Distribution of the maximum transit depth (dimensionless) for Exoplanets. Observations by the Kepler space telescope are shown separately. Solar system objects shown for comparison, data from Cox, 2000. [36] Data for exoplanets taken from Zolotukhin, 2017.[9]	21
Figure 2-6: (right) A plot of the radii and masses of low mass stars, brown dwarves and planets at 4×10^8 (red line) and 5×10^9 (black line) years from a model by Chabrier et al. [38] Four low mass stars (red circles) and Jupiter (blue triangle) are shown for comparison. Image: ESO [39].....	22

Figure 2-7: The geometric probability for alignment suitable for transit depends on orbital distance and stellar radius, assuming the planet to be small relative to the star. Formula and data for the solar system from Koch et al., 2005. [37] Data for exoplanets taken Zolotukhin, 2017.[9].....	23
Figure 2-8: Distribution of proportion of time in transit. Data for Exoplanets from Zolotukhin, 2017.[9] Data for Solar System from Koch & Gould, 2005. [37].....	24
Figure 2-9: Position and orientation of the Field of View (FoV) can maximise the number of reference stars. Images: Stephen O’Driscoll, Dept. of Applied Physics & Instrumentation, CIT. [2]	37
Figure 3-1: Spectrum of parallel computing	42
Figure 3-2: Schematic of the physical organisation of Grid Ireland.....	44
Figure 3-3: Conceptual structure of the grid from the user perspective.	45
Figure 3-4: Programming Language Paradigms	50
Figure 3-5: Project Software Strategy.....	59
Figure 5-1: Principles of operation of the Locus Algorithm for two stars.....	74
Figure 5-2: Celestial Coordinate System showing the convergence of meridians towards the pole and the consequent foreshortening of unit angles in Right Ascension. From A review of Coordinates, Redmond [129]	75
Figure 5-3: Overview of the Locus Algorithm Process. Data from the Catalogue is aggregated to form a Mosaic in memory. Candidate reference stars are identified within this Mosaic, and a Rating is calculated for each. A Locus is drawn around each where a FoV may be centred. The Points of Intersection (PoI) between these Loci are identified and a score calculated for each. The PoI with the highest score is output as the optimum pointing for that target. Each of these steps is illustrated in its respective subsection below.....	77

Figure 5-4: Modified image taken from SDSS Navigate image showing fields. [130] To obtain data on all stars in the green box, data from the SDSS fields marked in red must be aggregated to form a mosaic.	78
Figure 5-5: Reference stars must be identified from among the stars in the mosaic. The target is shown in white. Potential reference stars are indicated in green. Rejected stars are indicated in red. Stars are rejected if they are too bright or faint (indicated by size), if their colour indices are different to the target (indicated with 7 point stars) or if they cannot be resolved (indicated with overlapping stars.).....	79
Figure 5-6: The loci are defined by assigning a pair of RA and Dec coordinates to a cornerpoint and a pair of Boolean switches which indicate whether to draw a line North or South and East or West from the cornerpoint. Each Star is assigned a colour, and the locus that corresponds to it is drawn in the same colour.....	80
Figure 5-7: The intersection points between the lines are the points at which the score changes. For clarity, each star has been assigned a rating of 1 in this example. As a result, the score for each PoI is equal to the number of stars within a FoV of the PoI...	81
Figure 5-8: Locus Algorithm. Target: white star. Pointing & FoV: blue. Reference stars and their loci: Fully in the FoV: greens. On the edge of the FoV: yellows. Outside FoV: reds.....	82
Figure 5-9: Four conditions upon which a target will fail. Clockwise, from top left: (1) a target with no references, (2) a target with one reference, (3) a target the loci of whose references do not intersect and (4) a target for which the loci of its references are nested such that they do not intersect one another.	83
Figure 5-10: Potential rating systems. The binary scoring system, with a rating of 1 within the limit, and 0 outside. Under the triangular and square systems, rating trends from 0 at the limit to 1 at a perfect match. Triangular uses a linear progression, while square uses a parabolic progression.	85
Figure 5-11: The Witch's Hat graph. This graph demonstrates the distribution of rating against difference in two colour indices calculated as shown in Equation 5-6.....	87

Figure 6-1: Illustration of Top-Down design concept. The overall task is broken into components recursively until it can be coded. Note that not all components require the same number of steps to refine as shown in the case of Part 2 in the illustration above.	91
Figure 6-2: Illustration of Bottom-Up design concept. A number of components are created which are assembled into more abstract components and eventually used to form the overall program. This diagram also highlights one potential issue with Bottom-Up design. Components (e.g. parts g & i in this diagram) may be designed and developed but are unused in the final design. [134]	93
Figure 6-3: Top-Down design in breadth-first and depth-first modes. Breadth-first solutions follow the path indicated by the letters in the alphabetical order (I excluded for clarity.) Depth-first strategies follow the decimal pattern (e.g. 1.1 comes ahead of 1.1.1, which comes ahead of 1.2.) [139]	95
Figure 6-4: Design and Development Cycle	97
Figure 6-5: Software Design Document Example: an early version of the filtering process by which candidate reference stars are identified. The current version of this design forms the basis of Subsection 7.2.2.1.1. Note the informal notes in red text which indicate design decisions which were outstanding at this point in the development process.	99
Figure 7-1: Bottom-Up Perspective on the Software Design of this Project	108
Figure 7-2: Top-Down Software Design: User Parameters define SQL queries to the CAS and are used by the Parameterisation system to define API jobs which process the SDSS catalogue to generate the Local Catalogue. The Local Catalogue is then processed with further user arguments in the Main Data Pipeline to produce the Output Catalogues. Each component is discussed in its respective subsection. Not shown are Grid Management (7.2.4) and Error Checking (7.2.6) systems which are integrated within other software elements.	110
Figure 7-3: Data Access API	112
Figure 7-4: Diagnose	113

Figure 7-5: Extract	115
Figure 7-6: Operation of Pipeline in Target List mode.....	117
Figure 7-7: Operation of Pipeline in Catalogue Traversal mode	118
Figure 7-8: Call pipeline script (dashed boxes represent the modifications dependent on mode).....	119
Figure 7-9: Locus Algorithm Program.....	120
Figure 7-10: Filter Function	122
Figure 7-11: Locus Main Function	124
Figure 7-12: Parameterisation Modes. User input, together with the data contained in the CSV file from the CAS determines which mode parameterisation employs. Target List and API parameterisation run on and store their output on gridUI, while Catalogue Parameterisation mode is a grid job, managed by its own suite of GMS, storing the parameter files generated on the LFC.	126
Figure 7-13:API Parameterisation.....	129
Figure 7-14: Pipeline Parameterisation. Note that the target input module shown here is the one for target list mode. Figure 7-15 shows the changes made for Catalogue Traversal Mode	132
Figure 7-15: Target Input module for Pipeline Parameterisation in Catalogue Traversal mode. Note that as PRT files only include FITS paths, the target input module produces no output to the PRT file.....	133
Figure 7-16: Grid Management Software	135
Figure 7-17: SQL Queries to generate input for Parameterisation	136
Figure 7-18: Error Checking System	140
Figure 8-1: The data storage and processing elements used within the project, and the interactions between them.....	144

Figure 8-2 Top Level Directory Structure: This structure is used on all physical storage elements, and constructed as needed below the working directory on that element.....	148
Figure 8-3 Workspace Directory: The top level of subdirectories is used for overall version control. Lower levels are automatically generated by the Eclipse IDE.	148
Figure 8-4: Scripts Directory: Subdirectories are used for families of scripts.....	149
Figure 8-5: Test Directory: a partial directory tree for the test directory. A sample test (test 2 of 8-6-2011) is shown with an expanded directory structure. As can be seen from this example, a given test does not necessarily generate a full directory structure.....	150
Figure 8-6: Data Folder: The SDSS Catalogue structure is shown in Figure 8-7. The subdirectory structure of the parameter directories and the naming convention of the files are the same in the jdl, prm and ppr directories. Output catalogues in test, xop and qso directories are typically stored in a directory with their date, and may include further information in the directory or file name as shown in Table 8-1.....	151
Figure 8-7: SDSS Directory structure: As is shown in this diagram, the same directory structure is used for both Local Catalogue and Raw SDSS catalogue files.....	153
Figure 8-8: Definition of an SDSS <code>tsObj</code> File Path. Given Run (R), Rerun (r), Camcol (C) and Field (F): Rerun and Field in the filename are padded with leading zeroes to the length illustrated, but leading zeroes are not used in the directory names, nor are they used for Rerun or Camcol in either the filename or the directory. [113] An arbitrarily selected example is shown.	154
Figure 8-9 Release Folder	154
Figure 8-10: General Structure of a FITS File. Each fits file consists of one or more HDU. The HDU can be separated in to the Header and the Data Unit. The Header may contain zero or or more name-value pairs which describe the data in the Data unit. The data unit consists of zero or more rows each consisting of a series of column entries corresponding to the name-value pairs in the Header. The data in these columns may be a single entry or a vector of multiple values.	156

Figure 8-11 SDSS <code>tsObj*.fits</code> File: Only one HDU is used, which contains a mean of 847 rows and 146 columns of data. These columns contain entries which have either one value (referring to the observation as a whole) or five values, one for each SDSS band filter	158
Figure 8-12: Local Catalogue FITS File. The local catalogue files contain RA, Dec and a vector of 5 magnitude values for each star that passes the extract process shown in Subsection 7.2.1.2	159
Figure 8-13: Output FITS File: Output data included RA, Dec and an array of mag as per the Local Catalogue. In addition, the RA and Dec of the pointing (<code>av_ra</code> and <code>av_dec</code>), and the score for that pointing are given.	160
Figure 8-14: CTI File: each CTI file describes the contents of a FITS file. It lists the number of rows in the file as a long int, the number of columns as an int. It then has a repeating structure for each of the 146 columns in an SDSS fits file consisting of an int for the column number, and an array of 16 characters for the column name, and three more integers for the typecode, repeat and width of that column.....	161
Figure 8-15 API Parameter Files (PRM): API parameter files consist of a long int representing the number of files to be processed as part of the job the file represents, and an array whose entries are arrays of characters – the path to each file. The length of these paths is a variable, defined to be 150.....	162
Figure 8-16 Pipeline Parameter Files (PPR): PPR files consists of 4 observational parameters, and a repeating structure for each mosaic to be generated, itself consisting of two repeating structures, one listing the paths to the files to be aggregated into that mosaic, and the other listing the targets to be analysed within that mosaic. Each repeating structure has a long int controlling variable.....	163
Figure 8-17 Parameter Text Files (PRT): these files are ASCII text files used to list file paths without requiring binary file access. They consist of a line for each file, terminated with a newline character, and each line consists of a sequence of characters representing the path to that file.....	163

Figure 8-18: API CSV File: These files consist of lists of run, rerun, camcol and field for each of the fields to be processed. These numbers can be translated into file paths using the definition shown in Figure 8-8	166
Figure 8-19 Pipeline CSV File (Target List mode): These files consist of a number of sets of lines, each corresponding to the mosaic of fields around a target in the target list. Within these groups, each line corresponds to a field in the mosaic. Target information is repeated on each row within the group.....	167
Figure 8-20: Pipeline CSV File (Catalogue Traversal mode): These files consisted of a number of sets of lines, each corresponding to the mosaic of fields around a target field. Within these groups, each line corresponded to a field in the mosaic. Target field information was repeated on each row within group.....	168
Figure 8-21: Job Description Language (JDL) files: These files consist of a number of key-value pairs specified in strings with specific formatting determined by the <code>gLite</code> User Guide [7].....	169
Figure 10-1 Fields required for target star SDSS J113824.40+483457.8 (dark green). All areas which can be included in a FOV with the target are highlighted in bright green. The 14 fields which must be accessed to create this area are highlighted in red. Image taken from SDSS Navigate tool. [130]	182
Figure 10-2, There are 10,813 entries (shown in red) in SDSS among the 14 fields required for SDSS J113824.40+483457.8. Of these, 2447 met the criteria for a star (highlighted in orange.) Only the 1457 stars marked in green could be included in a FoV with the target (white with black cross).....	184
Figure 10-3: Filters to identify reference star candidates. The 42 candidate reference stars, highlighted in cyan boxes in the were selected from among all stars (green diamonds) within a FoV of the target (white and black cross) by including only those which were of the correct magnitude (white circles), and which matched the target on both g-r (magenta diamonds) and r-i (yellow squares) colour indices.....	185
Figure 10-4: Possible Intercepts. For a given corner point and locus (yellow) there are four relative positions other cornerpoints can reside at. From each of these, Loci can be	

drawn North or South and East or West. Loci that produce an intercept are shown in green, those that do not are shown in red.....	187
Figure 10-5 The Locus algorithm applied to all 42 candidate reference stars, indicated by blue diamonds. The Loci for each of the references are shown in red. The optimum pointing, and the Field of View centred on that point are highlighted in Green. The reference stars used in this optimum pointing are highlighted with yellow circles.....	188
Figure 10-6: Image taken from Raheny Observatory based on pointing for SDSS J113824.40+483457.8 (circled in red). Reference stars are circled in blue. Note that the FoV for Raheny Observatory is larger than the FoV for which the Exoplanet catalogue was originally intended. The size and position of the original FoV and pointing is approximated in a green overlay.	191
Figure 11-1: Distribution of scores in the Quasar Catalogue. This graph shows only the distribution of scores for quasars for which a valid pointing could be identified.....	199
Figure 11-2: Distribution of scores for a sample of 10^6 stars in the Exoplanet Catalogue. Not shown in this sample are 57105 stars for which no pointing was observed	202
Figure 12-1: Distribution of Magnitudes for a sample of 10^6 stars from SDSS. The proportion of the sample is plotted against the proportion of the distribution of stars in the whole sky as predicted by Allen. [157] Errorbars are not included as the Poisson error is too small to be displayed.	207
Figure 12-2: Distribution of scores as a proportion of overall targets in a given r magnitude bin. The addition of a variable offset allows for separation of the magnitude binning. Errors shown are given by Poisson statistics	208
Figure 12-3: Plot of failed targets against magnitude. Shown in grey in the background is the overall distribution of stars by r magnitude. Errorbars are not included as the Poisson error is too small to be displayed.....	209
Figure 12-4: Distribution a set of descriptive statistics of scores for a sample of 10^6 stars in the Exoplanet Catalogue, separated into magnitude bins and plotted against magnitude. Note that Maximum and 95 th percentile values are plotted on the left-hand	

y-axis, while other variables are plotted on the right axis as the former have a much wider range of values.	210
Figure 12-5: Colour Distribution of Stars in SDSS. All four colours using neighbouring magnitudes are shown here. No errorbars are shown, as Poisson statistical variation would not be visible at this scale.	212
Figure 12-6: Distribution of count of scores against score and g-r (left) and r-i (right) colour indices for a sample of 10^6 stars in the Exoplanet Catalogue. Stars for which no pointing could be determined are excluded from these plots. The distribution of scores against colour index can be observed to be similar to the distribution of stars by colour index as shown in Figure 12-5. The peak of each distribution is highlighted in green and red respectively.....	214
Figure 12-7: Proportion of targets in a given bin of colour index that failed to find a pointing, plotted against the proportion of stars in the sample in that bin.....	215
Figure 12-8: Distribution of a variety of descriptive statistics of scores against g-r colour index. Note that Maximum and 95 th percentile values are plotted on the left-hand y-axis, while other variables are plotted on the right axis as the former have a much higher range of values.	216
Figure 12-9: Distribution of a variety of descriptive statistic of scores against r-i colour index. Note that Maximum and 95 th percentile values are plotted on the left-hand y-axis, while other variables are plotted on the right axis as the former have a much higher range of values.	217
Figure 12-10: SDSS Navigate image for J203733.62+001953.5, the target with the highest score of any target in the sample of 10^6 stars from the Exoplanet Catalogue. Shown in green is the target, in red are each of the 247 reference stars selected for use with that target to produce a score of 117.70, in yellow is the pointing – the point at which the telescope should be aimed and in blue are the boundaries of a 0.25 degree field of view centred on that pointing. This image may be contrasted with Figure 10-6 to illustrate the increase in the number of reference stars for this “top” target.....	219

Figure 13-1: Distribution of Local Catalogue File creation times over 15 minute intervals from 25 th -27 th August 2010	235
Figure 13-2: Exoplanet Catalogue output files generated over time in bins of 1 hour	238
Figure 14-1: Effect of changing maximum colour index difference between target and reference star on the rating. A candidate reference star with a colour index difference of 0.5 is shown as a example. When the maximum permitted colour difference (ΔCol_{max}) is 0.5, its rating is 0. When ΔCol_{max} is 1, its rating is 0.5, and when ΔCol_{max} is 2, its rating is 0.75.....	247
Figure 14-2: Locus Algorithm with different FoV sizes. Target, references and pointing identical to Figure 5-8. Blue outline shows the original pointing. Purple and Magenta show larger and smaller FoV sizes respectively. Solid lines show results of using the originally calculated pointing. Dashed lines show the optimal pointings for those FoV sizes. Circled stars indicate those references affected by the size of FoV	249
Figure 14-3: Conceptual structure of the AWS from the user perspective. The similarity of this structure to the structure of the grid as shown in Figure 3-3 suggests that many components of the software could be reused should the project be translated into a cloud computing paradigm.....	254
Figure 14-4: Ratio of the spherical correction factor for the northern edge of a 15arcminute FoV to that for the southern edge. As is shown, these values are very close to equal at low-medium Dec.	259
Figure 14-5: Position and orientation of the Field of View (FoV) can maximise the number of reference stars. Images: Stephen O'Driscoll, Dept. of Applied Physics & Instrumentation, CIT. [2] (Duplicate of Figure 2-9)	261
Figure 14-6: a Sketch of the Spiderweb Algorithm in simulation using the same simulated starfield and colour scheme as in Figure 5-6. By drawing the possible orientations of the field such that one target is at the corner and another on the edge of the FoV, it is possible to identify an optimum pointing and rotation, shown in blue. This field includes 6 targets as opposed to the 5 shown in Figure 5-6.	262
Figure 15-1: Table of Text Styles	302

Figure 15-2: Programs & Scripts	303
Figure 15-3: Functions	303
Figure 15-4: Processes	303
Figure 15-5: Loops & Repeating Data Structures.....	304
Figure 15-6: Logical Operations	304
Figure 15-7: Computing Elements	304
Figure 15-8: Storage Devices.....	305
Figure 15-9: Stored Data.....	305
Figure 15-10: Catalogues	305
Figure 15-11: Data Flow	305
Figure 15-12: User Input.....	306
Figure 15-13: Developmental Tools	306
Figure 15-14: Documents.....	306

1. Introduction

This Chapter introduces the project in two major Sections. Section 1.1 introduces the fundamental concepts of the project and the project objectives in brief. The scope of the project and the techniques used to achieve the objectives are outlined.

Section 1.2 uses the Hierarchy of the Thesis as outlined in Section B-a to describe the Parts and Chapters of the Thesis and outline the overall structure of the document as a whole.

1.1. Introduction to the Project

This project uses novel Data Mining techniques to analyse large astronomical catalogues, harnessing the power of Grid Computing to produce new catalogues of pointings optimised for Differential Photometry for observers searching for extrasolar planets by the transit method.

Differential Photometry, as discussed in Section 2.2 is the measurement in changes in the flux of light from a source over time relative to other sources. When observing variable astronomical phenomena, this variability must be established by comparison with one or more reference stars. The more such reference stars, and the closer the colour match these reference stars are for the target of observation, the more precise the photometric observations that will result. [1]

It is apparent that adjustments to the position of the Field of View (FoV) to be observed can increase the number and quality of reference stars available for comparison. Most simply, the field of view may be moved so the target star is off-centre, but still within the field of view, to include additional reference stars. The challenge is posed by the choice of which adjustments to make. [2]

In this project, an automated method for making this decision was developed, called the Locus Algorithm. [3] The Locus Algorithm, defined in detail in Chapter 5, generates a *pointing* and *score* for each target, given a set of parameters. These pointings are pairs of Right Ascension (RA) and Declination (Dec) coordinates upon which to centre the field of view of the telescope, optimised to provide a maximum number of reference

stars which most closely match the target, and the score is a measure of how well optimised that pointing is.

This algorithm is applied to large astronomical catalogues to provide output catalogues of pointings for each star. This task is an example of data mining: extracting useful knowledge from large data sets. [4] The scale of current astronomical databases (e.g. the Sloan Digital Sky Survey (SDSS): 4.76TB of data for 357,175,411 objects [5, 6]) makes such a data mining task a computationally intensive process.

A distributed computing solution provides faster processing by breaking the overall task into discrete jobs which can be processed in parallel on multiple processors. Several distributed paradigms exist at present. Of these, grid computing was chosen for this project. Grid computing demands that jobs submitted using its own unique computing language, known as JDL. [7] In addition, appropriate software must be written to manage this submission process.

Two large catalogues, described in detail in Chapter 11, have been generated using these techniques: a catalogue of pointings for all quasars in SDSS; and a catalogue pointings for of all stars in SDSS.

This project brings together expertise from the fields of astronomy, astrophysics and computer science to develop algorithms to extract useful information from astronomical catalogues. These techniques have been informed by advanced photometric techniques developed by the Astronomy & Instrumentation Group at Cork Institute of Technology (CIT) and have benefited from the use of high-performance computing resources from Grid Ireland at Trinity College Dublin. (TCD)

1.2. Structure of the Thesis

The Thesis is broken up into four main Parts, framed by the Preface, which explains the conventions and terms used in the project, this Introduction and the Appendices, as illustrated in Figure 1-1. These four Parts are

- Part I: Background – Chapters 2-4 outline the fundamental domain and technique information required for the project

- Part II: Design & Implementation – Chapters 5-9 define the solution to the overall project problem, and describe the design and implementation of that solution.
- Part III: Results – Chapters 10-13 state and provide analyses of the results of the project from an astronomical and computational perspective.
- Part IV: Future Work & Conclusions – Chapter 14-15 propose a number of projects which are to be built upon the basis of this project, and conclude the Thesis by summarising the project as a whole.

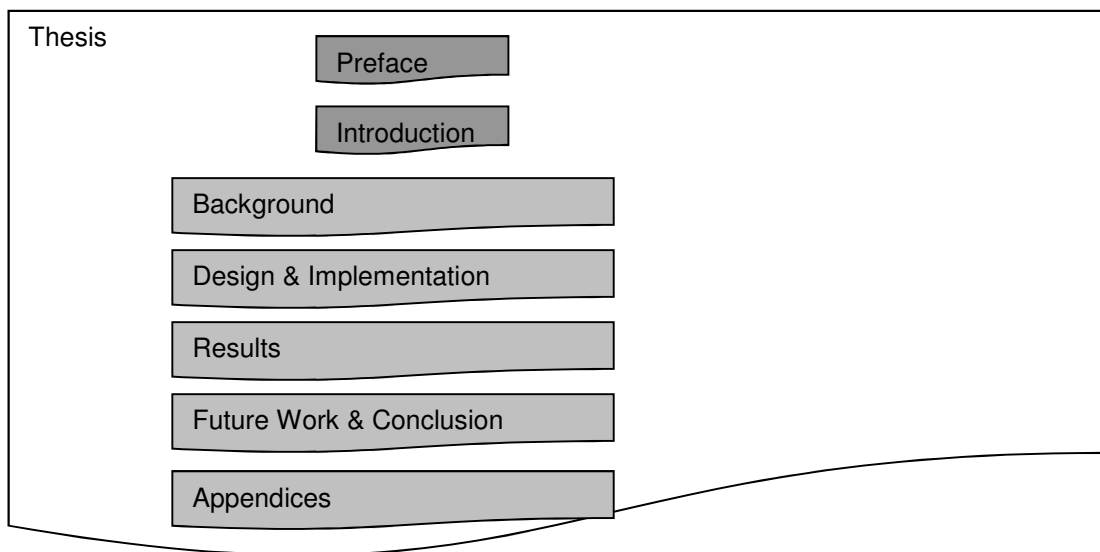


Figure 1-1: Thesis Structure

1.2.1. Part I: Background

Chapters 2-4 provide the background information upon which the rest of the Thesis is based. The background Chapters show *why* this project was undertaken. These Chapters, taken together, give a fundamental understanding of the concepts involved and the resources available and needed for this project. This Part of the Thesis demonstrates the scope of the project. Chapter 2 is primarily concerned with the known science of variable objects. Chapter 3 describes existing technologies such as grid computing. Chapter 4 states the objectives of the project - the contribution the project is designed to make to those fields. **Part I: Background** positions the project in the correct context, both historical and contemporary, for **Part II: Design & Implementation**.

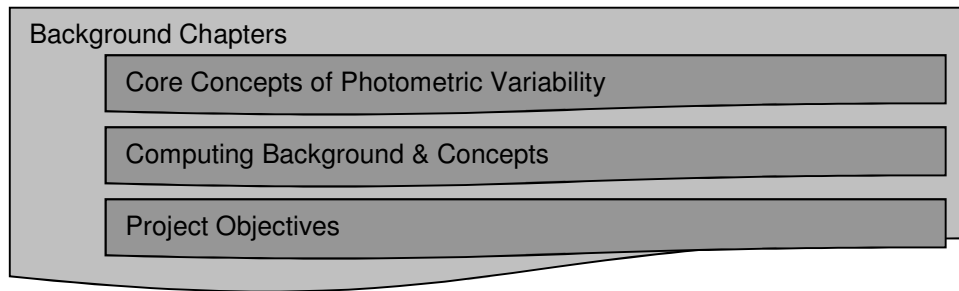


Figure 1-2: Background Chapters

1.2.1.1 Astronomical Background

In Chapter 2, the basic scientific information required to understand the project is introduced. This Chapter begins by introducing the concept of variable astronomical phenomena and how they can be observed. Here, a historical foundation is laid by describing the observation of variable stars, especially eclipsing binary stars. The earliest extrasolar planet observations are then considered, followed by a discussion of current trends and approaches in exoplanet detection and surveys. The techniques and technologies involved in Photometry are also discussed, again from a historical perspective first, then moving up to the modern techniques of Ensemble Differential Photometry.

1.2.1.2 Computing Background & Concepts

The next Chapter discusses resources required and available for this project. The computing requirements for this project are assessed in terms of software, hardware and data storage requirements. An overall view of High Performance Computing (HPC) is explained, providing context for the HPC solution which is used in this project, Grid Ireland. Chapter 3 describes the software components of the project, first defining overall concepts for the project, and then explaining the specific solutions chosen.

1.2.1.3 Goals of the Project

The last Chapter in this Part of the Thesis lays out the **Goals of the Project**. This Chapter is split into two Sections: the Astronomical Goals, and the Computational Goals. The Astronomical Goals Section defines the improvements that the project is intended to provide for differential photometry surveys. The need for a reliable algorithm and a means of comparing candidate stars is demonstrated. The requirement

to scale these techniques from single targets to large catalogues and adapt to a variety of parameters is highlighted. The Computational Goals are two-fold. First, the project must demonstrate a capability for data-rich, process-poor computing in a grid environment. Second, the project must provide numerical analyses of its performance in the form of metrics, defined here. Chapter 4 makes extensive reference to Part III: Results.

1.2.2. Part II: Design & Implementation

The second Part of the Thesis considers the Design and Implementation of the project. These Chapters move from a discussion of *why* the project was undertaken to describing *how* it was accomplished. This Part begins by defining the Locus Algorithm, the overall solution to the project, and describes the novel technique at the centre of this project. It then outlines the concepts and philosophy of design used in the project. Then, the design of the project is laid out from the top down to the fine details in two Chapters: Project Design, which considers the conceptual structure of the project and Data Storage and Management, which defines the data structures and file types used in the project. This Part of the Thesis is completed by a description of the practicalities of Implementing and Operation of the project.

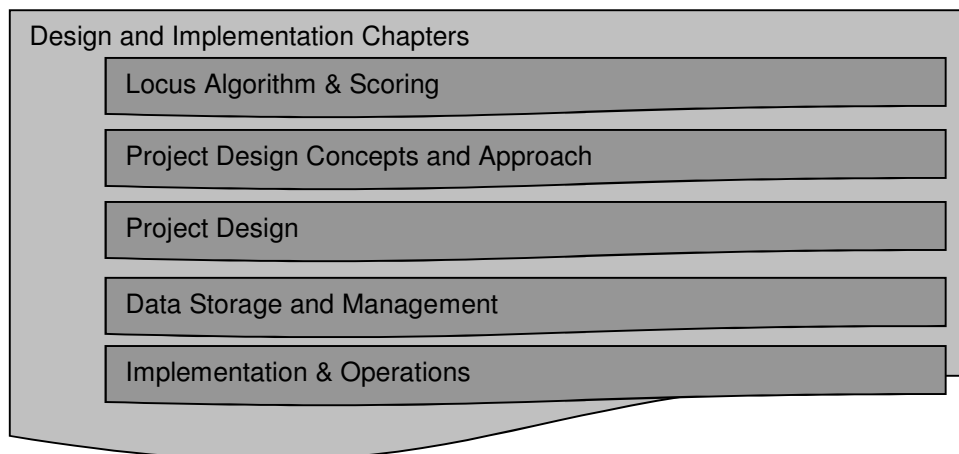


Figure 1-3: Design and Implementation Chapters

1.2.2.1 Locus Algorithm & Scoring

Chapter 5 describes the Locus Algorithm, the novel approach to providing optimised pointings which lies at the heart of this project. The algorithm is defined by means of a

step-by-step description of the process at the finest level. A key, modular component of the algorithm is the Scoring system. Several examples of scoring systems are shown. One of these was selected for the large-scale release versions used to produce the results shown in Part III. The rationale for this choice is explained, and that system is described in detail.

1.2.2.2 Project Design Concepts and Approach

Next, the design requirements of the project are defined, that is a framework by which multiple systems can interact over the course of the project. The design process is then shown: a step-by-step, evolutionary procedure with a robust system of version control which uses detailed design documents.

1.2.2.3 Project Design

Using the tools and following the philosophies defined in the previous Chapter, Chapter 7 shows the overall design of the project. Working Top-Down from an overall design, the Chapter steps through each component of that design, breaking them down to the pseudocode level. The three main components defined in separate Sections in this Chapter are the Applications Programming Interface (API) which abstracts the data from the sources, the Pipeline which uses this abstracted data to produce the outputs, and the Management software which ties manages the submission of jobs to the grid.

1.2.2.4 Data Storage and Management

A robust structure was developed to manage the data as described in Chapter 8. First, the data storage and processing elements of the project are defined. The second Section of the Chapter describes the data storage structure that is used throughout the project to ensure reliable access to the data in spite of the multiple storage systems used. Finally, each of the data types used in this project is described in detail. Both externally defined data types and novel data types defined for this project are used, and the definition of and the rationale for the use of each is given.

1.2.2.5 Implementation & Operations

The work cycle of development, coding and implementation used throughout the project is shown in the final Chapter in this Part of the Thesis. The ongoing operations of the

project are explained in detail. Grid operations in particular place boundaries on the project, and these are discussed here.

1.2.3. Part III: Results

While Part I: Background demonstrates *why* the research was undertaken, and Part II: Design & Implementation describes *how* it was carried out, Part III: Results shows *what* was achieved. These four Chapters discuss the output from the project, both from an astronomical and computational perspective.

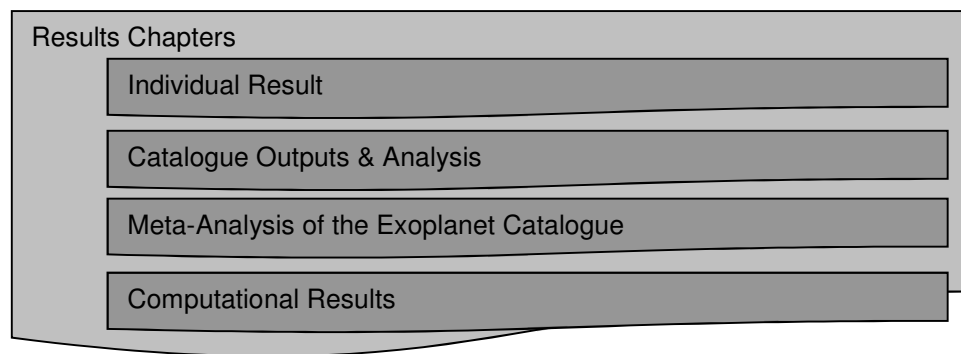


Figure 1-4: Results Chapters

Firstly, a fully described worked example of the project in action on a single target is shown. Next, the catalogues produced are described from an overall perspective. The results of the Exoplanet Catalogue are subject to a meta-analysis which discusses the trends and patterns shown in the data. Finally the computational metrics developed over the course of the project are explained, presenting the capability that has been demonstrated by this project.

1.2.3.1 Individual Result

This Part of the Thesis begins with Chapter 10. This Chapter consists of worked example of the algorithm in action on SDSS J113824.40+483457.8 a star selected based on observational constraints at Raheny Observatory. The results of an observation of this star from Raheny Observatory are presented. This result consists of the output from a single instance of the project's operation: each of these outputs would form an entry in the catalogues produced.

1.2.3.2 Catalogue Outputs & Analysis

Chapter 11 presents the catalogues produced over the course of the work. Two separate catalogues are shown: a catalogue showing pointings for all quasars in the SDSS footprint and a catalogue showing pointings for all stars in SDSS for exoplanet observation. The rationale for each of these catalogues, and an explanation of the capability this demonstrates is given. The parameter set used to create each of these is also stated. Short excerpts from these catalogues may be found in the Appendices.

1.2.3.3 Meta-Analysis of the Exoplanet Catalogue

This Chapter presents a meta-analysis of the exoplanet catalogue produced from a phenomenological perspective. This meta-analysis shows trends observed within the data, and allows an end-user to interpret catalogue results. Notably, this process includes identifying patterns in scoring, so the user can identify high and low scores relative to the overall distribution of scores.

1.2.3.4 Computational Results

The concluding Chapter of this Part of the Thesis, Chapter 13, shows quantitative assessments of how the project operated. Three main areas in which the project can be judged are discussed, Data Metrics – measurements of data volume and composition, Processing Metrics – the time and processing power taken to complete various project components and Grid Metrics – measurements of the performance of the grid solution by comparison with non-grid computing solutions.

1.2.4. Part IV: Future Work & Conclusions

The last of the four Parts of the Thesis consists of a Chapter on Future Work and the Conclusion. The Future Use & Refinements of the Project Chapter describes several projects to be built on this project. The Conclusion Chapter summarises the project and explains in brief how the objectives were met.

Having described *why* to carry out the project, detailed *how* it is carried out, explaining *what* has been discovered, the final Part of this Thesis describes *where* these results can be taken in the future.

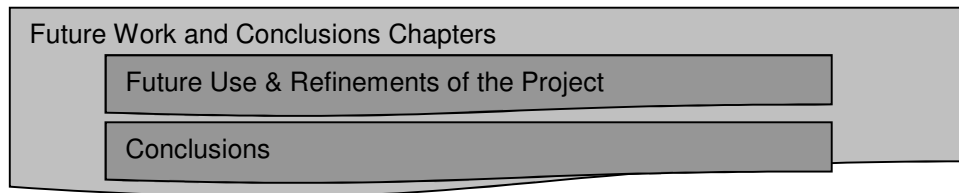


Figure 1-5: Future Work Chapters

1.2.4.1 Future Use & Refinements of the Project

This Chapter suggests Future Use & Refinements of the Project that might be undertaken. The large parameter space of the software and the limited utilisation made of that space in the course of this project is discussed. This Chapter describes possible explorations of this parameter space. These include suggestions for other classes of object to study. The use of the designed flexibility to allow the project to operate with forthcoming catalogues is discussed. Several modifications to the software are proposed in this Chapter. Finally, additional software functions beyond those used in this project are proposed.

1.2.4.2 Conclusions

The Conclusions Chapter provides a brief assessment of the project as a whole, and summarises the output of the project, explaining in brief the objectives of the project, and stating briefly how those objectives have been met.

1.2.5. Appendices

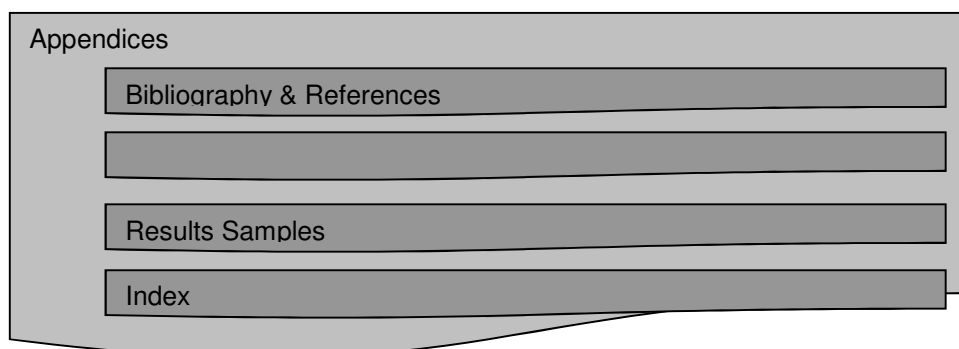


Figure 1-6: Appendices

There are four appendices to this Thesis. The first Appendix is a bibliography and references, which compiles full citations to all of the documents referred to throughout

the Thesis. Second is a series of code excerpts demonstrating the layouts and styles used in the various languages while programming this project. Next, a set of extracts from the results of the project is presented. The last Appendix is an index.

1.3. Summary

This Chapter introduces the project in brief, outlining the astronomical problem which the project is intended to solve, and summarising the solution that has been developed to do so. A brief outline of the structure of this Thesis is provided, explaining the function of each Part of the document, and highlighting the roles the individual Chapters play in the overall work.

2. Core Concepts of Photometric Variability

This Chapter introduces the core concepts of astronomical photometric variability by exploring the classes of objects to be studied, and the techniques used to study them.

The Chapter begins by considering the class of object to be studied: variable objects. A brief introduction to astronomical variability is given. This introduction consists of a definition of what is meant by astronomical photometric variability, and a reference to the broader context of the observation of variable astronomical objects.

The Chapter then discusses several specific classes of variable astronomical objects in detail. Three primary classes of object are considered: Variable stars, which were the first variable objects to be studied after the planets of the Solar System; [8] extrasolar planets, which are most commonly observed by the effects they have on their parent star; [9] and finally quasars, a very different class of object which may be observed with similar techniques to those used to observe exoplanet transits. [10]

The next Section of this Chapter discusses the techniques used to observe these phenomena. Any measurement of the amount of light that is detected from an object can be considered a photometric observation. The Section is broken into three Subsections. The first Subsection, 2.2.1 defines three overall classes of photometric observations used: Absolute, Relative and Differential. The second, 2.2.2, discusses the historical development of photometric techniques which may be used to generate those observations. Next, in Subsection 2.2.3 there is a detailed consideration of the specific operational parameters of the technique used in this project: differential photometry using CCD-based observations of ensembles of stars. [10, 11]

Finally, Section 2.3 discusses the use of astronomical catalogues to identify stars which meet user-defined astrometric and photometric criteria.

2.1. Variability of Astronomical Objects

This project is concerned with the photometric variability of astronomical objects. Photometric variability, as discussed in Section 2.2, is change in the amount of light detected from an object. [12]

Objects as varied as rotating asteroids [13] and Active Galactic Nuclei (AGN) [14] exhibit photometric variability. Some variability is intrinsic to the object being observed. Other variations are extrinsic: it can be attributed to external causes. The variability can be dramatic, as in the case of Mira, the first confirmed variable star, which has been observed to vary from 10.1 to 2.0 in Johnson V Magnitude, [15] to minor variations such as that of the Sun as it rotates with sunspots on its surface (about 0.01 in V magnitude over about 30 days) [15]. There can be once-off events (like supernovae) or regular cycles (such as eclipsing binary stars.) [8] They can take place over long timescales (up to the lifetime of a star) [16] or short timescales (millisecond pulsars) [17].

A full treatment of all variable objects is beyond the scope of this project. Instead, this Section discusses three classes of objects.

- Variable stars were the first of these three classes of objects to be observed. The history of the observation of variable stars and several causes of stellar variability, both intrinsic and extrinsic are discussed in brief. One class of variable star, that of eclipsing binary stars, is of particular significance in this project. This is because observations of eclipsing binary stars can be considered a precursor to the observation of extrasolar planets by the transit method as discussed in Subsection 2.1.1. An understanding of the history, techniques and technologies used in studying these objects can thus help guide the study of extrasolar planets.
- Extrasolar planets, also referred to as exoplanets, are planets which orbit a star other than the Sun. The history of the observation of these planets, and a number of techniques used to observe them, are discussed in Subsection 2.1.2, beginning with the 1992 observation of PSR1257+12b by Wolszczan & Frail [18] and the 1995 observation by Mayor and Queloz of 51 Peg b [19]. The particular technique this project focusses on is the transit method, pioneered in 2000 by Charbonneau et al [20] and Henry et al. [21] This method is defined and discussed in detail in Subsection 2.1.2.1.

- Quasars are much more luminous, and much more distant objects, than the nearby stars observed as exoplanet host candidates. They are described in Subsection 2.1.3. [14] However these objects, too, show small-scale visible-wavelength variation on a range of time scales. [22] These variations can be observed using similar techniques to those used to study exoplanets. [10] [23] · These similarities in observational techniques are described in brief in Subsection 2.1.3.

2.1.1. Variable Stars

Aristotle, whose writings influenced other classical philosophers, believed that the cosmos was unchangeable: stars did not, indeed could not vary [16]. This belief was challenged by observations of cataclysmic events such as novae and supernovae wherein a star would suddenly "appear" in the night sky, only to disappear again. The modern interpretation is in fact that all stars vary over the course of their lifetimes. [24] Moreover, it is thought that all stars vary to some extent over shorter timescales: for example the Sun is known to vary by about 0.1% in Johnson V magnitude over the 11-year Schwabe cycle. [25] Both of these classes of variability are beyond the scope of this Thesis, which is concerned with variations over a shorter timescale.

The first surviving records of any recurring variable star are those of Mira (Omicron Ceti), observed by Fabricius in 1596 and 1609. Halwarda later measured the period of variability for Mira to be 11 months, in 1638. This can be considered the starting point of the known history of variable stars. [26, 27]

Algol (Beta Persei), is a star whose variability may have been known since ancient times, as suggested by mythological associations such as that with the Gorgon in Greek mythology. [27] The first definitive records of its variability are Montanari's from 1667. Its periodicity was measured by Goodricke in 1783, who later proposed a mechanism for its variability – that a darker body regularly passed between the Earth and Algol – a proposal that later observations would confirm. [28] This mechanism is functionally identical to that used to observe variability in a star due to an exoplanet transit, as discussed in Subsection 2.1.2.1.

Calculations of the system's parameters were made by Pickering in 1881, and largely confirmed by Vogel's 1889 measurement of its radial velocity variations by means of the Doppler effect. [8] From these measurements, amongst others, Algol is now known to be a close binary star system with a third component in a much wider orbit. [16]

The two close components: a brighter star, Algol A and a fainter companion, Algol B, form a system where each passes between the Earth and the other at various points in its orbit. This causes a reduction in the light from the system, as some of the light from the more distant of the two is obscured by the closer star. In the case of Algol, this reduction can be observed with the naked eye. [28] Such a system is referred to as an eclipsing binary system, and many such systems are now known.

Smaller scale variation than that observed in Algol can be measured using photometric techniques as discussed in Subsection 2.2.2. A series of observations of a photometrically variable object can be plotted to form a light curve as discussed in Section 2.2. In the case of an eclipsing binary star, several key parameters of the lightcurve can be determined, from which parameters of the system can be calculated by fitting simulated models to the observed lightcurve. The following parameters are relevant to this Thesis, as these same parameters can be measured for extrasolar planets.

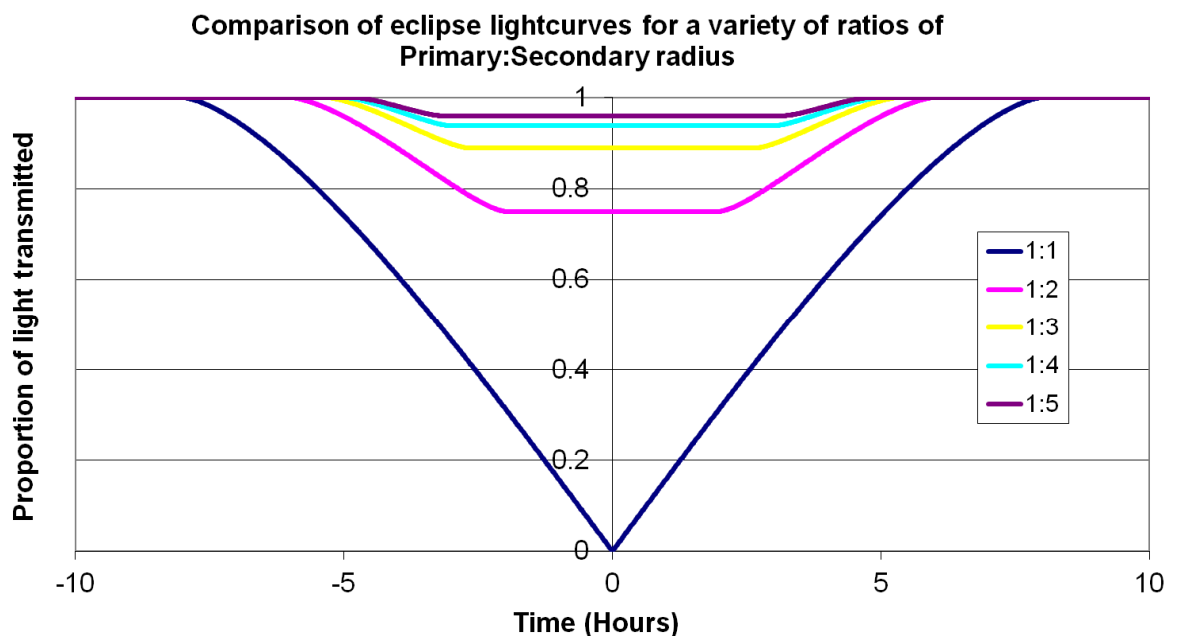


Figure 2-1: Calculated duration, shape and depth of light curves for uniformly dark, centrally eclipsing secondary objects of various radii orbiting uniformly illuminated primary objects at constant velocity.

- **Depth:** Transit depth is the degree to which observed brightness is reduced. Assuming a central eclipse, a uniformly dark transiting object and a uniformly illuminated primary, this would depend solely on the proportion of surface of the object which is obscured. The maximum value of this degree of obscuration would thus give a direct value for the ratio of cross-sectional area of the two objects, as shown in Figure 2-1. In practice, additional factors such as luminous secondary objects, limb darkening and star spotting can complicate this calculation for stars. [21] Improved photometric precision permits shallower transits to be observed, as well as revealing details of the shape of the lightcurve as discussed below.
- **Period:** the eclipses repeat over a cycle equal to the orbital period of the objects. Deviations from this period indicate a more complex system: e.g. a third companion, whose presence may be confirmed by other measurements. This is the case with as Algol C, whose presence changes the timing of the eclipsing binary pair Algol A and B. [29] Improved temporal precision permits subtler effects on eclipse timing, such as those caused by exomoons, to be observed. [30]
- **Duration:** The duration of the eclipse is related to the apparent path length taken by the obscuring object across the surface of the object being obscured and the velocity of the orbit. In addition, larger secondary objects will cause obscuration for longer durations than smaller ones as shown in Figure 2-1. As with period, improved temporal precision permits improved measurement of this quantity.
- **Shape:** The shape of the lightcurve can vary, depending a number of factors. A larger eclipsing object will take longer to reach the minimum brightness than a smaller one as shown in Figure 2-1, while a grazing eclipse will show a different shape of lightcurve to a central eclipse as demonstrated by Figure 2-2. In addition, the shape of the lightcurve can reveal the effects of limb darkening. [31] Improved precision in photometry permits the shape of the lightcurve to be more precisely defined, which permits improved discrimination between, for example, grazing transits of stars and central transits of exoplanets.

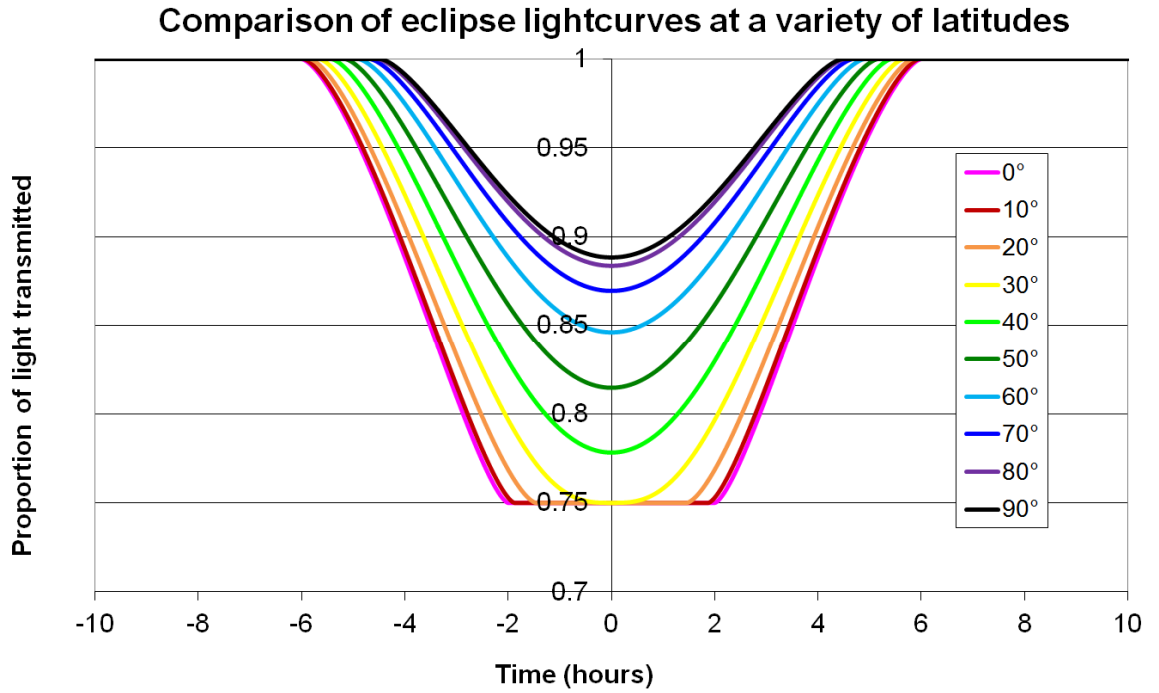


Figure 2-2: Calculated duration, shape and depth of light curves for uniformly dark secondary objects of constant radii ($r_{\text{secondary}} = 0.5R_{\text{primary}}$) transiting the primary at different apparent latitudes.

A number of additional parameters can be measured by observing the parameters above.

- **Orbital geometry:** The shape and orientation of the orbit of the two stars determines whether an eclipse occurs, as the eclipsing body must cross the narrow area of space between the Earth and the more distant body, indicating a low angle between the orbital plane and the line of sight to the Earth. The calculated transit latitude of the obscuring body can be used to calculate this angle more precisely.
- **Secondary Eclipse:** For a given system, a secondary eclipse may also be observed, where the brighter object obscures the fainter one, and for which the same parameters may be measured. The fainter the secondary object is, the smaller this effect is, and as a result the more sensitive the photometry needed to observe a secondary eclipse. As orbits may be eccentric, sometimes highly so, the presence of a primary eclipse does not guarantee that a secondary will be observed, nor vice-versa.
- **Radial Velocity:** Spectroscopic measurements can be used to supplement the photometry by calculating the radial velocity of the components of the system.

These measurements allow for calculations of the relative movements of the objects, Measurements of the radial velocity variation of eclipsing binary stars can prove exceptionally useful as the inclination between the orbital plane and the line of sight is known to be near zero for an eclipsing system. Thus, the radial velocity is known to be the full velocity of the objects, and thus can be used in the calculation of their relative masses.

2.1.2. Exoplanets

Planets are much less massive than stars, and, by definition, do not generate their own energy by nuclear fusion – the current working definition of an exoplanet is an object with a true mass below the limiting mass for thermonuclear fusion of deuterium (~13 Jupiter Masses for objects of solar metallicity) that orbits a star. [32] This means they are necessarily much fainter than their host stars, and the angular separation between star and planet is typically small. [17] This makes direct observation difficult. Indirect techniques, however, have proven more successful as demonstrated in Table 2-1. This Section discusses these techniques.

Number of planets detected	Detection method	Proportion of detections
698	Radial Velocity	19.57%
76	Direct Imaging	2.13%
2703	Transit	75.78%
1	Astrometry	0.03%
51	Microlensing	1.43%
24	Pulsar Timing	0.67%
5	Transit Timing Variation (TTV)	0.14%
9	Other	0.25%

Table 2-1: Breakdown of number and proportion of planets detected by various techniques. Taken from the Interactive Extra-solar Planets Catalog, Zolotukin, 2017. Retrieved 25th January 2017 [9]

The first confirmed extrasolar planet was observed by timing variations of the pulsar PSR1257+12 in 1992 by Wolszczan & Frail. [18] In 1995, a planet was discovered orbiting the main sequence star 51 *Peg* by Mayor & Queloz, using the radial velocity method. [19] Both of these observations made use of the fact that the planet and its host star orbit their mutual barycentre, which leads to variations in the timing of pulses from the pulsar, and the radial velocity of the host star. Since then, there have been 3,567 confirmed extrasolar planet discoveries as of the 24th of January, 2017. [9] A breakdown of these detections is given in Table 2-1. The techniques used to observe

these planets are described in brief below, summarised from The Exoplanet Handbook, Perryman, 2011. [17]

- **Radial Velocity:** The movement of the star about the barycentre can be measured using the Doppler shift of its spectral lines. This method measures variations in the velocity component of the star along the line of sight, but gives no information about transverse movements. The absence of information about a transverse component means that radial velocity observations alone cannot distinguish between a low mass planet in an edge-on orbit and a higher mass object in an almost face-on orbit.
- **Transit:** This technique detects the reduction in light from a star when a planet passes between the earth and the star, analogous to the observation of eclipsing binaries above. This is the technique that this project is focussed upon, and it is discussed in detail in Section 2.1.2.1.
- **Direct Imaging:** Stars are between 10^5 (in the infrared) and 10^{10} (in the visible) times more luminous than planets, and the angular separation of a planet from its host star is typically no more than a few arcseconds for nearby systems. These facts impose technical challenges on direct imaging requiring very high signal to noise and resolution.
- **Timing:** The signal from any system with a measurable temporal signal (e.g. an eclipsing binary star system or a pulsar) can be disrupted due to changes in the light travel time caused by the orbit of the primary about the barycentre.
- **Gravitational Microlensing:** From general relativity, it can be understood that mass distorts space-time, deflecting the path of electromagnetic radiation. When a large mass passes between the observer and a distant source, that source can be magnified. When an exoplanet host system passes between the observer and a distant source star, complex, time-dependent behaviour can be observed as a result of the time-dependent changes in alignment.
- **Astrometry:** Extremely high resolution imaging can measure the position of a star to sufficient precision to track its transverse movement due to the gravitational influence of a planet.

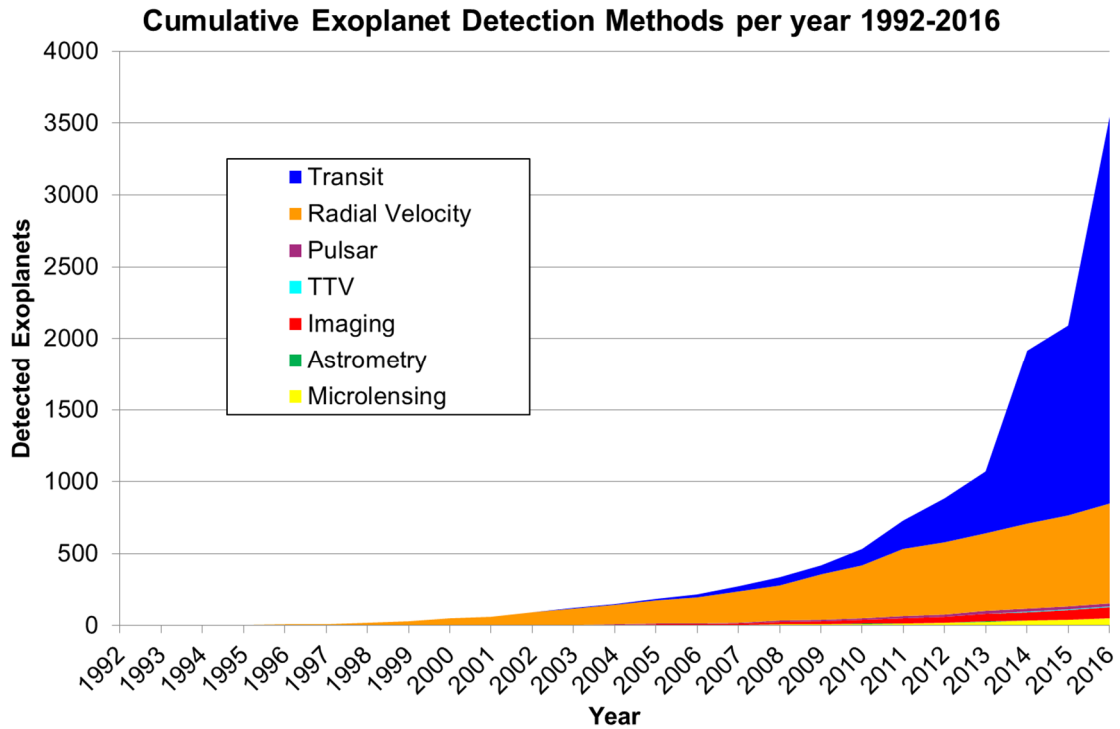


Figure 2-3: Exoplanet Detections by method as a function of time. Taken from the Interactive Extra-solar Planets Catalog, Zolotukin, 2013-2017. Retrieved 25th January 2017 [9]

These methods can be used to complement one another, and such complementary observations can be necessary to confirm the object is observed is, in fact, a planet. For example, a planet detected by the transit method must be subject to follow-up measurements using the radial velocity method to calculate the mass of the object - the lightcurve from a low mass star eclipse and a large planetary transit can be similar. Astrometric observations in combination with radial velocity measurements can provide a 3-dimensional projection of the orbit of the star about the barycentre. With such a projection, the parameters of the system can be fully calculated. [17]

Exoplanet observations can require highly specialised equipment. Direct Imaging and Astrometry require extremely high resolution imaging which is beyond the capabilities of all but the largest and most advanced ground-based telescopes with expensive adaptive optics, or space-based observations. The Radial Velocity method requires precise spectroscopy, which demands specialist equipment. Gravitational Microlensing depends upon chance alignments which are non-recurring events. [17] Transit observations, on the other hand, can be made with relatively inexpensive equipment, including small aperture (as low as 0.2m) telescopes and consumer grade CCDs. [33]

2.1.2.1 Transit Method

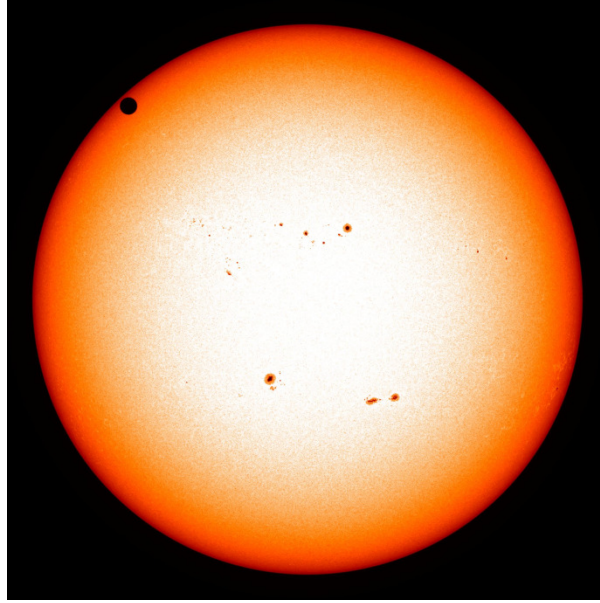


Figure 2-4: (left) An image of Venus in transit across the sun, demonstrating a planetary transit and limb darkening in the Sun. Image: NASA/SDO, HMI [34]

The Transit method is a photometric method for observing exoplanets which works by observing the reduction in photometric intensity of a star as a planet passes between the observer and the star, in a manner analogous to eclipsing binary stars discussed above. Transits were first observed independently from 1999-2000 by Henry et al. (2000) [21] and Charbonneau et al. (2000) [20].

Due to the fact that most planets are much smaller than their parent stars, photometric sensitivity on the order of millimagnitudes is required to detect planets [17]. The limitations and requirements this places on the photometry are discussed in Section 2.2, but as discussed by Billings [35] and Castellano, [33] the detection of giant planets orbiting small stars is within the capabilities of modest equipment.

Transit depth is a dimensionless quantity used to measure of the reduction in observed light from the host star. This reduction can be measured by plotting a light curve as discussed above. Large planets orbiting small stars can easily be understood to give deeper transits. Photometric precision greater than the transit depth is necessary for a transit to be detected and precision substantially greater than the transit depth is

necessary to show the shape of the transit lightcurve as shown in Figure 2-2 above. This causes a selection bias in favour of larger planets orbiting smaller stars. [17]

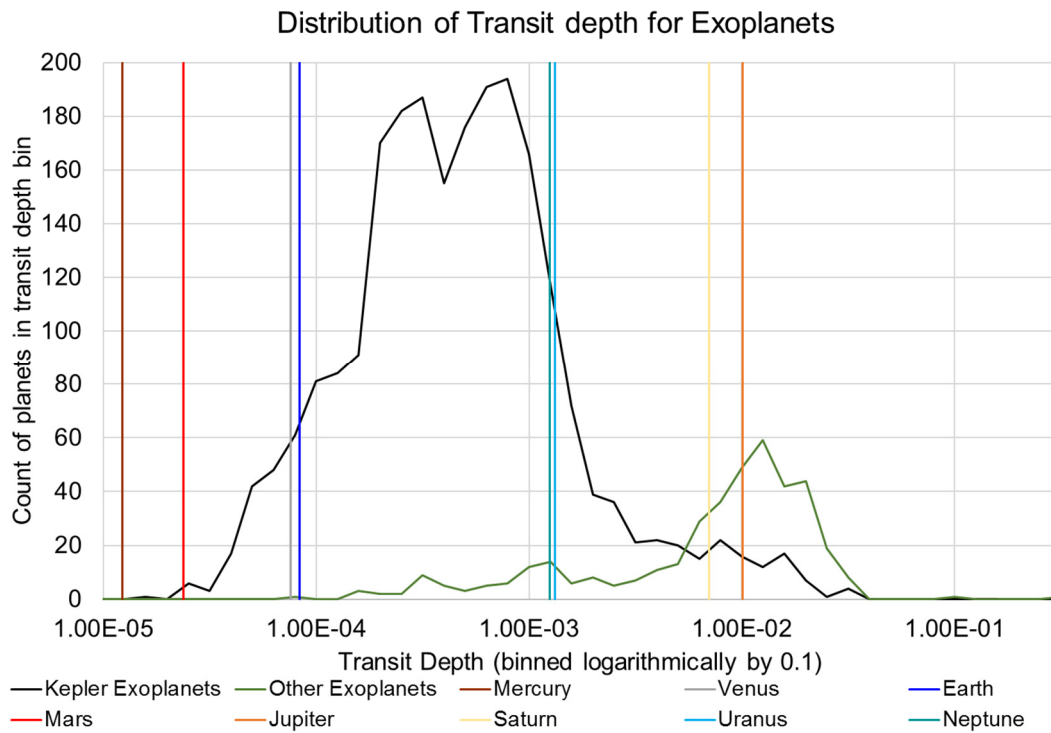


Figure 2-5: Distribution of the maximum transit depth (dimensionless) for Exoplanets. Observations by the Kepler space telescope are shown separately. Solar system objects shown for comparison, data from Cox, 2000. [36] Data for exoplanets taken from Zolotukhin, 2017. [9]

As can be seen in Figure 2-6, large planets can be comparable in dimensions to small stars, even though their mass is much smaller. As a result, the majority of exoplanets detected by ground based telescopes show transit depths comparable to or greater than those that would be predicted for the Solar System's gas giants, as shown in Figure 2-5. The transit depth for a perfect Earth analogue is 8.39×10^{-5} . [36]

Transits can only be observed when the alignment of the planet, its host star, and the observer is correct. This is only possible if the planet's orbit is aligned edge-on to the observer, to within the precision of the angle subtended by the star from the planet. The probability that a planet's orbit will be aligned correctly thus depends on the distance the planet is from the star, and the radius of the star itself, and can be determined geometrically.

Figure 2-7 demonstrates the geometric probability of a planet being in an alignment suitable for transits to be observed at various distances. For a perfect Earth analogue, the probability of alignment suitable for observation is calculated to be 0.47%. [37] Planets close to their parent star, and which orbit a large parent star are more likely to be suitably aligned.

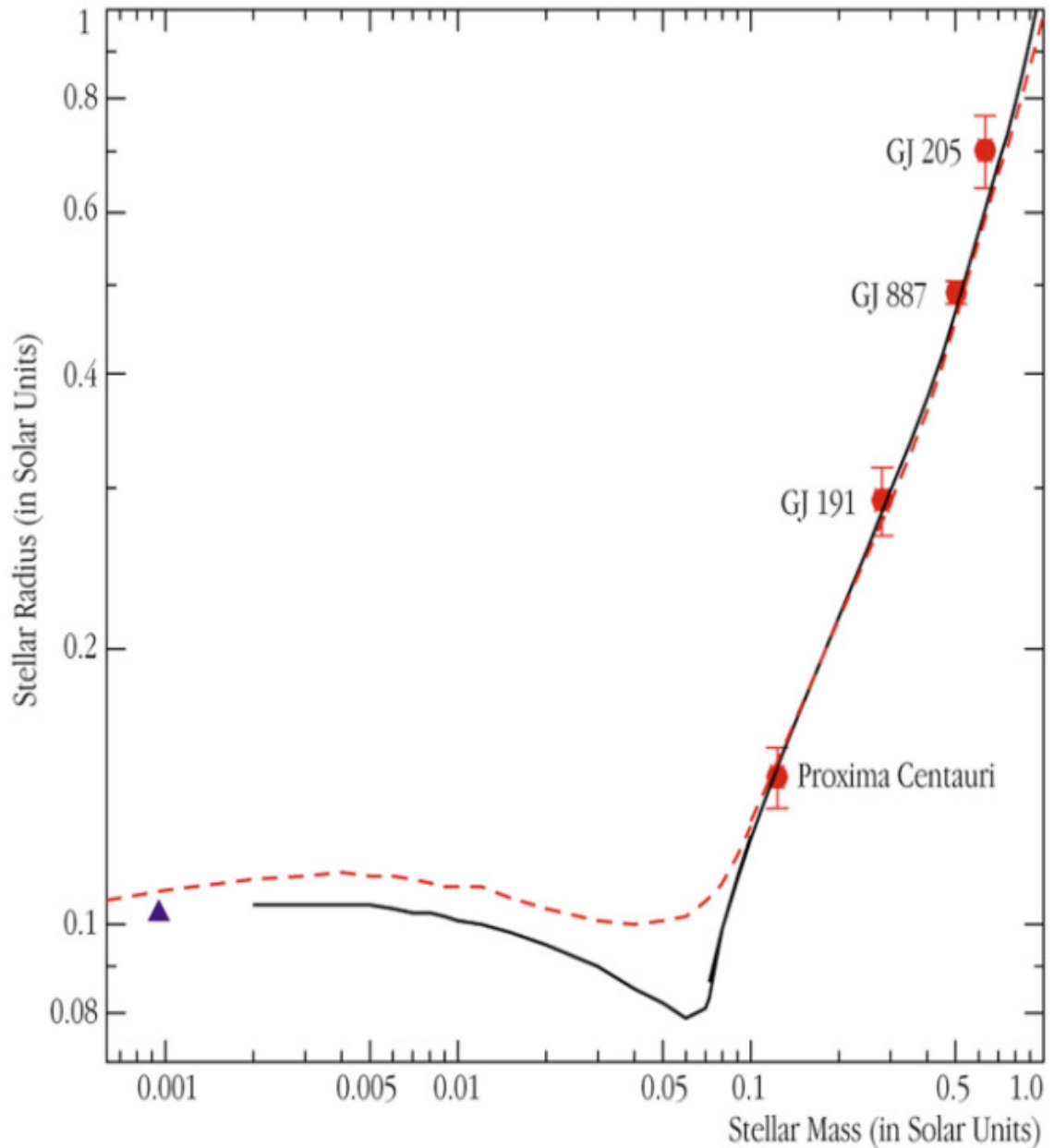


Figure 2-6: (right) A plot of the radii and masses of low mass stars, brown dwarves and planets at 4×10^8 (red line) and 5×10^9 (black line) years from a model by Chabrier et al. [38] Four low mass stars (red circles) and Jupiter (blue triangle) are shown for comparison. Image: ESO [39]

In addition, transits can only be observed for a short proportion of the orbit during which syzygy (3-body linear alignment) between its host star, the exoplanet, and the Earth occurs. This period can be calculated as the time it takes the planet to sweep out that angle of its orbit that corresponds to the angle that the star subtends in its sky.

For a perfect Earth analogue, a transit would last thirteen hours over the course of a full orbit, i.e. one year. This means that an Earth analogue would spend only 0.15% of its time in transit. [37] This again creates two further observational biases in favour of planets in close orbits: planets in close orbits are in transit for a larger proportion of their orbits, and those transits recur more frequently.

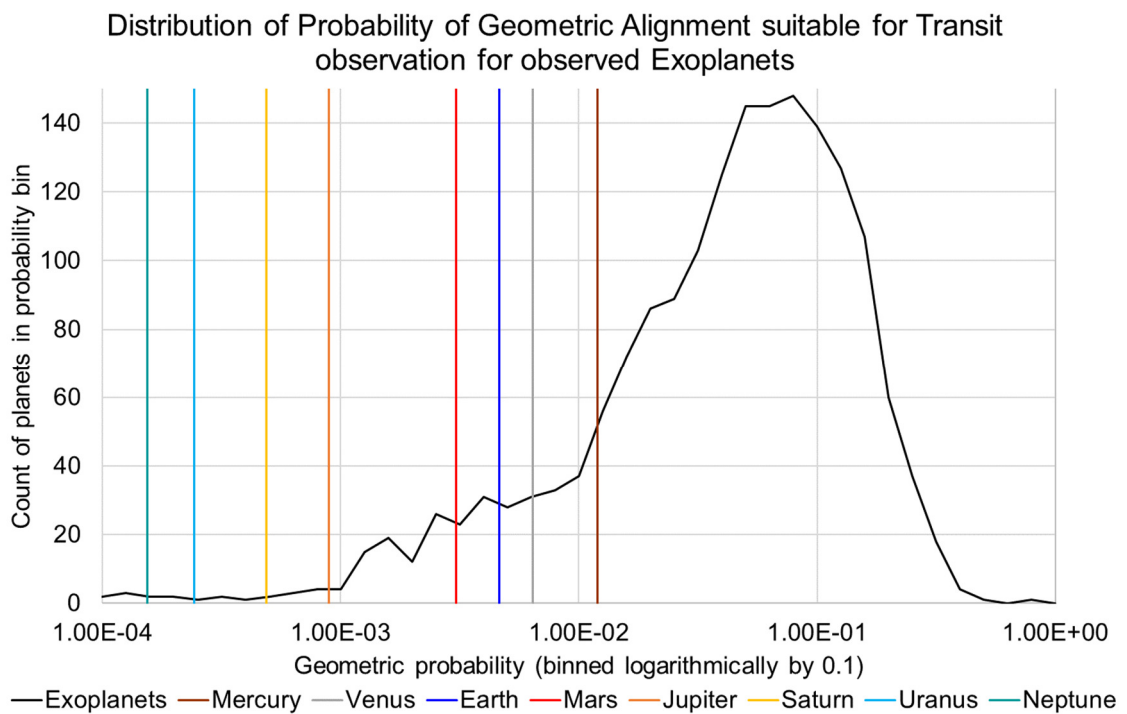


Figure 2-7: The geometric probability for alignment suitable for transit depends on orbital distance and stellar radius, assuming the planet to be small relative to the star. Formula and data for the solar system from Koch et al., 2005. [37] Data for exoplanets taken Zolotukhin, 2017. [9]

A transit survey, especially one using modest equipment, must take account of these selection factors. Increased photometric precision allows for smaller planets to be observed. A survey of as many stars as possible maximises the opportunity for a transit to be observed, due to the low probability that a transit is taking place at any given moment. As will be shown in Section 2.2, these goals are not incompatible.

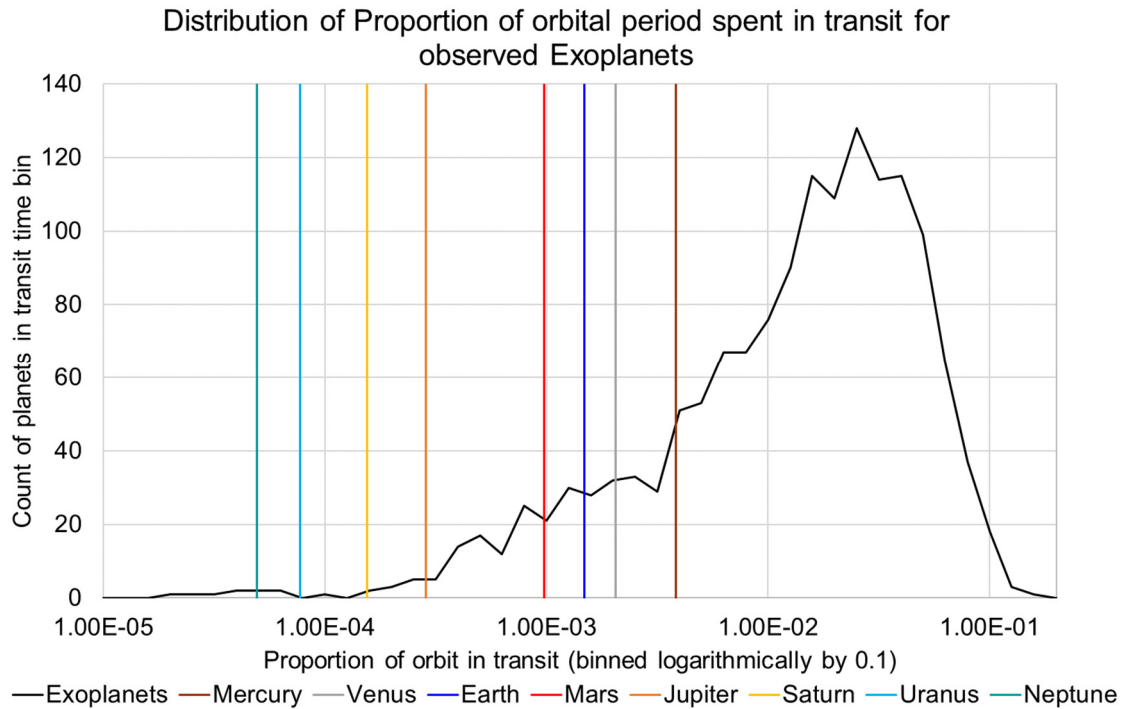


Figure 2-8: Distribution of proportion of time in transit. Data for Exoplanets from Zolotukhin, 2017. [9] Data for Solar System from Koch & Gould, 2005. [37]

2.1.3. Quasars

The term “quasar” comes from “quasi-stellar radio source.” The “quasi-stellar” part of the name comes from the fact that quasars appear as point sources even with the most advanced interferometers currently available. [22] Quasars are highly energetic, very distant active galactic nuclei. [14] They are powered by the accretion of matter onto the central supermassive black hole of the host galaxy. [40]

Observations of quasars show variations in both optical emission on short timescales. [41] However, the speed of light sets an absolute limit on the rate of change of any object: an object cannot change more rapidly than its size divided by the speed of light. For this reason, the phenomena responsible for short timescale photometric variations must occur over a small region.

By plotting a photometric light curve, an observer may probe regions which are too small to be resolved. The techniques used to observe and plot these light curves are similar to those used to observe transiting exoplanets. [10] These photometric techniques are discussed in detail in Section 2.2.

133,336 quasars had been recorded as of 2010, many of which were observed by the Sloan Digital Sky Survey (SDSS). [42] This is much smaller than the 357,175,411 unique objects recorded in the seventh SDSS data release, which formed the basis of this project. [43] As a result of this smaller dataset, and the similarities of techniques used to observe quasars to those used to observe exoplanet transits, it was decided to perform an analysis of the quasars in SDSS as discussed in Subsection 4.1.2.1 to test the software.

2.2. Photometry

Photometry is, at its simplest, the study of the brightness of stars and other astronomical objects. [12] In this Section, photometry in general is discussed under two primary headings. The first Subsection, 2.2.1, gives the definition and description of three major classes of photometry – Absolute, Relative and Differential. The second Subsection, 2.2.2, discusses the history of various photometric techniques, going from the naked-eye observations of ancient times to modern CCD-based ensemble photometry.

The third Subsection, 2.2.3, discusses the principles of operation of the photometric technique that this project is intended to work with, which is differential photometry. This Subsection considers the limitations placed by instrumental, environmental and astronomical factors on the accuracy and precision of the measurements needed to observe the phenomena discussed in Section 2.1. In addition, this Subsection describes how one might mitigate or eliminate these effects.

2.2.1. Classes of Photometry

Photometry may be considered in terms of three basic classes, although as Landolt notes, there is some disagreement in the literature as to the exact definitions of these classes. [44] This Subsection defines the following terms as they are used throughout this Thesis.

- Absolute Photometry measures the total apparent brightness of an object by reference to some standard. Landolt defines absolute photometry narrowly, accepting only standards based on spectrophotometry or with reference to a laboratory source such as a black body cavity. [44] Sterken & Manfroid favour a broader definition whereby absolute photometry may be defined in terms of a

set of standard stars such as the Landolt stars. [45] [46] The apparent brightness of Vega (Alpha Lyrae) has traditionally been used as a zero point for the magnitude scale, although its use has since been discredited, as it is itself variable. [47] [48]

In either case, the main source of error when carrying out absolute photometry is in the transformation to the standard. [49] This error can be as high as 5%. [50] As a result, absolute photometry is not suitable for this project, as this error is larger than the effects to be measured, and it will not be discussed further.

- Relative Photometry is the comparison of a target star with some comparison object. Landolt defines relative photometry as any comparison with a reference, including those with photometric standards. [44] Milone and Pel refer to any photometry which involves the comparison of two or more stars as differential photometry. [1] For the purposes of this Thesis, a distinction is drawn between measurements of the relative flux between two or more objects at a single point in time, which are referred to as relative photometry, and ones which have a temporal component, which are referred to here as differential photometry.
- Differential Photometry, as used within this Thesis, may be understood as a time-dependent series of relative photometric measurements. It is assumed that time-dependent atmospheric effects will apply similarly to objects which are close to one another. [10] Close matching between the target and the reference stars can maximise this correlation of effect. [51]

In the case of both Relative and Differential photometry, the traditional method was to compare the target star with a single reference, which was known or believed to be non-variable, known as the comparison star. [11] If a comparison star could not be identified which was known to be constant, a third star, known as the “Check star” was used to determine if any variability observed was due to the Target or the Comparison star, in a technique known as Comparison-Check (CK) photometry. [11]

The use of many reference stars, rather than just one or two, a technique referred to as Ensemble Differential Photometry, is favoured in modern differential photometry, as each can act as a check on the others, and greater redundancy can be achieved in

excluding a variable reference. This technique also allows for many targets to be assessed simultaneously. [50] This technique was made possible by the emergence of photometric CCD cameras as discussed in Subsection 2.2.2.5. [52]

2.2.2. Historical Development of Photometry

The history of photometry begins with ancient astronomers making observations with the naked eye, and crudely classifying the stars they saw into magnitude classes, which were later to evolve into the mathematical scale we still use today.

A mathematical scale demands instruments capable of making measurements which can be used in the calculations. This Section discusses the history of these measurements from Ancient Greece up to the modern day, when computer-controlled apparatus is designed for ever greater precision.

2.2.2.1 Visual

As described by Herbig (1945), [53] the magnitude scale, still used in a refined form today, is based upon the classes developed by Hipparchos (c. 127 BC) [54] which grouped stars by their apparent visual brightness. Ptolemy's seminal catalogue, *Almagest*, (c. 150AD) [55] is based in part on these observations. This ancient scale was simply a categorisation of stars from bright to faint. As can be seen from the fact that many dissimilar objects are in the first magnitude of this system, it was not intended as a robust mathematical scale. [56]

Many later observations used this or a related scale. Notably, for the purposes of this Thesis, Goodricke (1783) made his measurements of Algol by reference to whether it was brighter or fainter than neighbouring stars. [28] Using the definitions in Section 2.2.1, each point on Goodricke's measurement may be considered a primitive form of relative photometry, and the whole observation can be treated as an early version of differential photometry.

This type of observation, which relies upon both perception and memory, can give remarkably good precision, but very poor accuracy. [56]

2.2.2.2 Mechanical/Visual

The human eye is quite capable as a comparative instrument. Improved accuracy can be achieved if the fallible element of human memory is removed. [56] Starting with Bouguer (1760), instruments were developed which reduced the brightness of some bright standard by extinction, defocusing or demagnification until it matched that of a star. [1] [57] By measurement of the extent to which the standard object was reduced, the ratio of brightness of the star and standard could be established to a precision of between 2% [56] and 5%. [1]

Based upon measurements of this type, Pogson (1856) proposed the modern definition of the magnitude scale with a ratio of 100 in brightness between stars separated by 5 magnitudes. This proposal was based on an approximate fit to existing measurements. [58] [59]

2.2.2.3 Photographic

In principle, photographic measurements should be less subjective than visual ones: a physical record exists of the observation which can be revisited later. Bond (1859) developed an equation to relate the exposure time and the size of the image of a star. [60] Photographic measurements of objects of different colours depend on the response of the photographic emulsion.

These differences, and the later development of photographic emulsions that better mimicked the response of the human eye, led to the development of separate visual and photographic magnitudes. Again, however, the precision of these measurements was far greater than its accuracy, with mean maximum precision of ~2%. [56] [1]

2.2.2.4 Photoelectric

Photoelectric measurements of starlight began with Monck (1892) from his house in Earlsfort Terrace [61] and Minchin (1895). [62] [56] These early observations allowed a measurement of the intensity of light incident on the apparatus relative to a calibration. With a similar instrument, Stebbins (1910) made observations with a precision of about 2%, and was able to observe the secondary eclipse of Algol (i.e.

when Algol B is obscured by Algol A). [63] [56] The precision of these and later observations of a single target is limited by extinction variations. [1]

Big improvements could be achieved by applying differential techniques. [1] Walraven (1952) developed an instrument which would observe short-period variables quasi-simultaneously with a nearby reference star. [64] This quasi-simultaneity was achieved by “chopping” between the target and the reference. The use of a nearby reference star reduces the effect of extinction variations. [51] Photoelectric measurements depend upon careful selection of reference stars such that the references themselves do not vary on the timescale of the experiment. [51]

2.2.2.5 CCD

Charge Coupled Devices (CCDs) were invented at AT&T Bell labs in 1969 by Boyle and Smith (1970). [65] Initial CCDs had large instrumental errors, but with modern CCDs this has been greatly reduced. [66] [13] CCD photometry was established as being at least equal to photoelectric in 1984 by Walker, and has since surpassed its predecessor. [67] [65]

CCD photometry has advantages over photoelectric photometry in two key ways. The two-dimensional nature of the CCD means that the need to switch the observation between target and reference can be eliminated, as instead both objects can be observed simultaneously. This gives truly simultaneous data on target and reference stars. [52]- The second is that many reference stars, not one, (or two as in the C-K method) is possible, a technique called ensemble differential photometry. [11]

Ensemble differential photometry creates for two further advantages: first, the ensemble provides an ideal standard against which to perform differential calculations even if some members of the ensemble show low-amplitude variability. Second, all stars in the ensemble can be monitored simultaneously, and any variation in each can be measured. [11] This makes it an ideal technique for exoplanet transit surveys, as it can provide precise photometry of many stars simultaneously. [50]

2.2.3. Differential Photometry Operations

This Subsection discusses differential photometry in detail, under four principal headings. First the mathematical principles underlying differential photometry are explained, with particular reference to the increases in precision available by means of ensemble differential photometry. Secondly, a number of data reduction techniques are discussed in brief. Third, the sources of noise and the limits these place upon the precision of photometric measurements are discussed. Finally, techniques that may be used to improve upon these limits are discussed in brief. It is the automation of these improvements upon which this project is founded.

2.2.3.1 Mathematical principles

At its most fundamental, relative photometry calculates the difference in magnitude between the target star and a comparison star. Calculating the “true” magnitude of each would demand a transformation to a standard photometric system. As previously discussed, this greatly reduces the available precision.

$$\Delta m' = -2.512 \log \left(\frac{\text{object} - \text{sky}}{\text{comparison} - \text{sky}} \right)$$

Equation 2-1: The basic arithmetic task of relative photometry. Taken from Budding and Demicran [13]

However, for objects near to one another, both spatially and in terms of brightness, this transformation would be approximately the same. The net effect is that the transformation term cancels, and the result is given by Equation 2-1. [13]

This calculation will give the difference in apparent magnitude between the target star and the comparison. [13] This calculation alone, however, demands an idealised comparison star: Any variation measured in Δm is potentially traceable to either star, although comparison stars are usually chosen to be non-variable within a desired precision and timescale. [1]

The addition of a third star against which the comparison star is checked allows for a guarantee that both check and comparison stars are constant within the required precision so long as the ratio between check and comparison remains constant, as any

variation in either is unlikely to be correlated. [13] If that ratio does vary, the comparison process may need to be extended to a fourth or subsequent star until a stable comparison star can be found.

2.2.3.2 Ensemble Differential Photometry

Ensemble differential photometry may be considered the logical conclusion of the extended process of using a comparison and check star. In the simplest form of ensemble photometry, the total intensity of the ensemble is assumed to be constant, and any difference between the ensemble and the target star can be attributed to the target. [11] A more sophisticated model allows for an iterative process whereby each star in the ensemble is compared against the ensemble, and apparent variables are removed. [68] [69]

2.2.3.3 Data Reduction

Photometric analysis requires that flux data be extracted from an image of the star which may cover several pixels. The best statistical accuracy is provided by Point Spread Function (PSF) fitting, but this method is cumbersome. [65] More common is the use of aperture photometry which measures the instrumental flux within a circular aperture, and the sky background using an annulus around the star using technologies such as the Image Reduction and Analysis Facility, known as IRAF. [68] [70]

IRAF, first developed in 1981 at Kitt Peak National Observatory “is a general purpose software system for the reduction and analysis of scientific data.” [70] It provides a series of scripts, tools and tasks which are used in a variety of astronomical roles from the spectroscopy of comets [71] to the subject most relevant to this project, Ultrahigh-Precision CCD Photometry. [68] More recently, systems to overcome the shortcomings of the Command Language for IRAF as a scripting language have been developed. These systems, such as PyRAf, first released in 2001, have provided interfaces to the use of IRAF with the capabilities of more modern programming systems – e.g. Python in the case of PyRAf. [72] [73]

Using these interfaces is common practice for observational astronomers to develop automated pipelines to extract information that is relevant to their specific observations.

[73] Two such pipelines are of interest to this project: the SDSS Imaging Pipeline, which produced the data used as input for this project, [74] and `qvar`, a system developed at BCO Labs to apply the techniques of differential photometry to quasar observations. [22] [75]

The SDSS Imaging Pipeline gathers data from the 2.5m Apache point observatory telescope and pipes this information into a variety of data processing activities: for example: `Astroline`, which reduces the data from the telescope to manageable levels for further processing by creating “postage stamp” star cut-outs; and the `Frames` pipeline, which produces corrected frames, atlas images and, most relevant to this project, object catalogues. [74]

`qvar` automatically extracts the flux, creates a reference (“master-star”) from the ensemble, and analyses the data from each star, excluding references from the ensemble if they show variation of their own. The master-star can then be compared against the target to provide a differential photometry observation of that target. [22] [75]

2.2.3.4 Noise & Photometric Precision

In this Subsection, the causes of noise within a photometric measurement are discussed in detail. For each class of noise, the physical cause is addressed, and factors which influence its magnitude are discussed. In addition, systematic errors such as transformation errors may exist for a given measurement, but careful selection of comparison stars can reduce or eliminate this. [1]

2.2.3.4.1 Extinction Noise

Extinction is a reduction in brightness of an object caused by the absorption or scattering of some of the light by the intervening medium. Extinction noise, as applied to astronomical photometry, usually refers to unpredictable variations in atmospheric absorption and scattering. [76]

Extinction noise may be split into two subclasses: first order extinction (independent of colour) and second order (colour-dependent.) First order extinction depends primarily on airmass. If the difference in airmass is low, as is the case for stars close together, such as those in a small FoV, the difference in first order extinction is low. [1]

Burdanov et al. characterised this assumption by suggesting that the best photometry was available if the target and references were within a radius of 5-7 arcminutes of one another. [77]

Second order extinction is colour dependent: that is to say it occurs when certain wavelengths of light are preferentially absorbed and/or scattered by the atmosphere. This can be a major source of error in differential photometry with broadband filters. [51] In this case, different stars may have different distributions of light across the bandpass. If certain of these wavelengths are reduced but not others, a dimming will be observed more strongly in those stars which emit more of that light.

However, the effect of this can also be minimised if the target and reference stars are similar in colour. [1] Milone and Pel note that colour indices may be more useful than spectral type matching if, for example, there are substantial differences in interstellar reddening between the two stars. [1] This is because the significant factor in determining the utility of a reference star is whether atmospheric effects apply equally to it as to the target, not whether it is physically similar.

2.2.3.4.2 Scintillation

Scintillation is a random variation in the apparent brightness of a star, commonly referred to as “twinkling” for visible stars, caused by the refraction of starlight through heterogeneous pockets of air temperatures in the upper atmosphere. [10] This refraction causes light to be bent towards and away from the aperture at various points in time. [78]

Scintillation noise is proportional to the signal and dominates for bright stars. [1] Scintillation amplitudes are dependent on telescope size and exposure duration. Noise decreases with telescope size, as the refraction may deflect the path of the light onto a different part of the same detector, but larger telescopes are much more expensive. [51] Increasing the exposure time decreases the contribution of Scintillation to the overall noise budget. [78]

While scintillation continues to be a significant contributor to the limits of current ground based photometry, novel techniques using new generations of fast CCD's and real-time data analysis using smart algorithms can improve things significantly. An

example of this, of relevance to this project output and results, is a novel automated technique called SelPhot, which can mitigate the effect of the varying atmosphere to be removed by selecting only those exposures where the effect of the atmosphere is minimised, using a technique called “Lucky Imaging,” then using the technique of “shift-and-add” to stack these images on top of one another to provide significant improvements in photometric precision and resolution. [10] [79] [80]

Lucky exposures, first described by Fried (1978) are most readily acquired for telescopes with apertures (D) a few (3.5-4) times larger than the spatial extent of the atmospheric turbulent cells (r_0), and become highly improbable above 8 times larger than the cells. [80] This enables near-diffraction-limited imaging for small- (0.36m) to medium-sized (1.52m) telescopes. [81]

2.2.3.4.3 Photon Noise

Photon Noise, variously referred to as shot noise, white noise [1] or “Gaussian noise associated with Poisson counting statistics of photons,” [78] is a term for random noise inherent to counting photons from a variety of sources. When extracting the flux for a star from a CCD frame, both star and sky measurements can vary, both are subject to this source of noise, and this means both contribute to the overall noise budget. [1]

Photon noise obeys Poisson statistics – this means that the signal-to-noise ratio for any photon measurement depends on \sqrt{N} . [1] The relative contribution from Photon noise can therefore be reduced by increasing photon count. This can be achieved by increasing aperture or integration time, or by choice of a brighter reference star. [10] [1]

2.2.3.4.4 Detector Noise

Detector noise is noise generated by the detector and can come from several different sources. Detector noise occurs either in the act of absorbing the photons or in the process of reporting the detection of those photons. Which source dominates depends on the nature of the detector and the wavelength regime to be observed.

Detector noise may be driven by the quantum efficiency (QE) of the device, whereby some, but not all photons incident on the device are reported. This QE is often

wavelength dependent, but was more dominant in the past, as modern CCDs have very high QE. [44]

Thermal noise, whereby the detector itself emits light in the spectrum to which it is sensitive, is a major source of error in infrared photometry, but is less significant for instruments in the visible spectrum. [1] [82] Thermal noise can be treated by cooling the detector with liquid Nitrogen or Helium as needed. [1] [51]

Read noise is noise generated in the process of extracting data from detections in the device. In the case of CCDs, this is caused by the fact that CCD detections take the form of small electrical currents. They are therefore vulnerable to electronic interference. Early CCDs had very high read noise, but according to Howell, this problem was rapidly reduced with improved technology, and is no longer a dominant contribution. [65] Young and Milone & Pel suggest that amplifier noise was significant in the photoelectric era, but is now relatively insignificant. [51] [1]

2.2.3.5 Improvements to Precision

As can be seen from the above Subsection, and as Burdanov et al. demonstrate empirically, precision of differential photometric measurements can be improved by a number of methods. [77] Some of these methods are observer and/or instrument based, and are thus beyond the scope of this project. Others, however, can be addressed in advance by careful choice of reference stars, and the use of ensemble differential photometry with those reference stars. For example, photon noise on the ensemble, which is proportional to \sqrt{N} , would be expected to decrease in proportion to the square root of the number of photons from the references in the ensemble. [1]

Individual reference stars should be an approximate match for the target star in terms of magnitude and a close match in terms of colour. [1] [51] Additional reference stars provide more reliable photometry by permitting the elimination of variables from the ensemble and the reduction of relative errors by combining the measurements from multiple references. [69] [22]

This project focusses on addressing each of these requirements, by providing, for any given target, the optimised pointing with the most reference stars, most closely matched

to the target in terms of Magnitude and Colour, by shifting the point of aim of the telescope slightly to allow for more reference stars to be included in the FoV with the target.

In addition, for the purposes of the search for extrasolar planets, all reference stars in an ensemble can be compared with one another, to determine if any of the references are themselves variable. If they are, they are eliminated from the ensemble for the purposes of use as a comparator, but the variance in that reference can be tracked separately, and it can be determined whether it play host to an exoplanet. [69] [22]

The scoring system discussed in Section 5.3 provides a modular, user-controlled solution which incorporates these factors into a score for a given pointing. As discussed in Subsection 14.3.2, this system would be suitable for further in-depth analysis to provide improved precision in the scoring system.

2.2.3.5.1 Magnitude Match

Ideal reference stars are similar in magnitude to the target star, or slightly brighter. [1] This improves photon noise contributions to the overall noise budget. A reference star much brighter than the target runs the risk of saturating the detector. A sufficiently bright reference may even demand the use of a neutral density filter, which introduces yet another source of noise. [1] [51] A fainter one will necessarily show a large signal to noise ratio and may not achieve the constancy required of a reference. [1] [13]

2.2.3.5.2 Colour Match

It is widely advised that a reference star be close in colour to the target star. [1] [13] [51] Colour match may be achieved using existing photometric measurements of sufficient accuracy. Young (1991) suggests a limit of 0.3 mag difference in Johnson B-V colour index. [51] The SDSS catalogue, as discussed in Section 2.3, provides precise photometric measurements of 357,175,411 [5, 6] objects in five different colours. [43] SDSS colour indexes are used to rate individual reference stars in this Thesis as discussed in Subsection 5.3.1.

2.2.3.5.3 Point of Aim

In addition to careful selection of reference stars, it may be possible to get more, or better references by moving the centre of the field of view of the telescope. All potential reference stars in the vicinity of the target must be considered, and the field of view adjusted to include the best of these. [23] [2]

Care must be taken when performing these adjustments not to place a reference star or the target star too close to the edge of the field of view, as edge effects on the detector may cause difficulties. [83] [51]

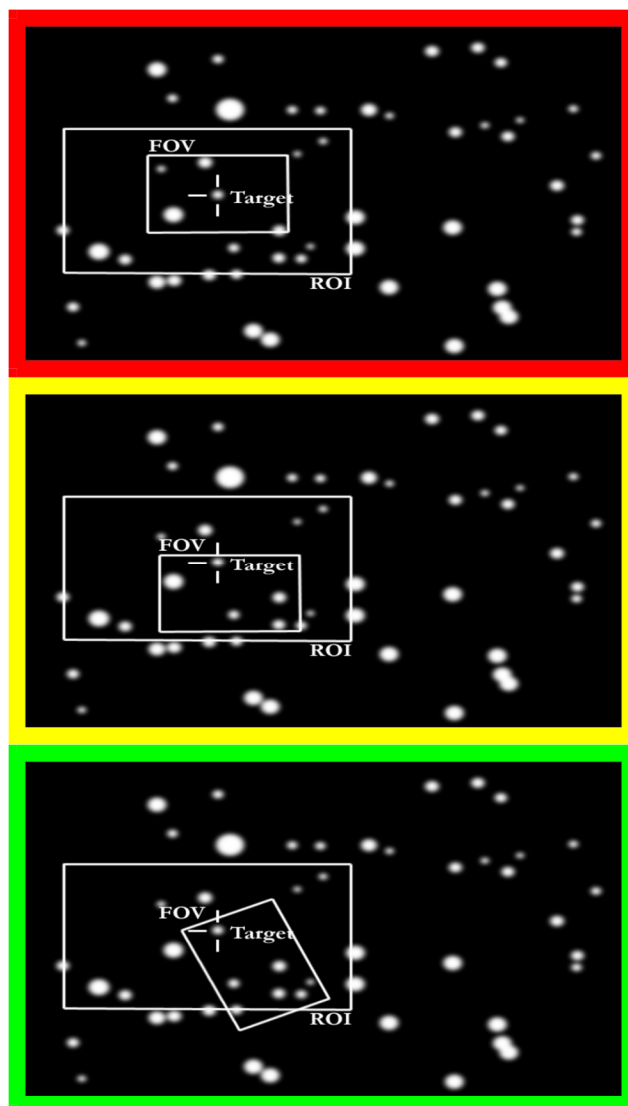


Figure 2-9: Position and orientation of the Field of View (FoV) can maximise the number of reference stars. Images: Stephen O'Driscoll, Dept. of Applied Physics & Instrumentation, CIT. [2]

A computationally simple approach is to simply translate the centre of the field of view on the North-South, East-West axes. A more thorough approach would also allow for rotations to the field of view, but that can be computationally intensive as is discussed in Subsection 14.3.1.3.

This project is based upon these principles: determining, for each of many targets, the best possible reference stars, and the correct telescope pointing to observe the target and as many of them as possible.

2.3. Application of Data Mining Techniques to Astronomical Catalogues

This project is designed to perform data analysis on astronomical catalogues to accomplish the improvements to precision described above. Astronomical catalogues consist of lists of many astronomical objects, usually with astrometric and photometric information on each object. Some catalogues, such as the Sloan Digital Sky Survey (SDSS) include additional information.

This photometric information allows for reference stars to be selected for closer match to the target in terms of magnitude and colour as discussed above. The astrometric information permits the identification of stars that can be included in a FoV with the target with an appropriate point of aim.

This allows the observer to plan observations in advance to ensure the best choice of reference stars. In addition, a target may be selected from amongst otherwise equally viable candidates based on the quantity and quality of reference stars available to the observer.

This project makes use of the SDSS Catalogue of Calibrated Objects (usually referred to in this project as the “SDSS Catalogue” or the Source Catalogue for brevity) for this purpose. As of 2006, during the planning phases of this project, SDSS was the largest photometric and spectroscopic survey in the optical wavelength range. [84] As such it was the most rigorous test available for the computing solution developed for this project at the time of development.

Subsequent catalogues, such as the Large Synoptic Sky Survey (LSST) are expected to surpass it in the coming years, and are considered as potential targets for future projects derived from this one as discussed in Subsection 14.1.5. [85]

2.4. Conclusions

Astronomical photometry is the study of the flux of light from an astronomical object. Astronomical phenomena vary over a variety of timescales, and by differing quantities. This project is concerned with variability on the timescale of minutes to hours, and of the order of 1% or smaller flux variations, such as are encountered with transiting exoplanets. This demands precise photometry.

Noise, dominated by the atmosphere, makes direct measurements of this precision impossible from the ground. However, the atmospheric effects on the flux from similar stars which are close to one another are similar. Therefore, by observing two or more stars at the same time, the atmospheric effects can be reduced by comparing the change in relative flux over time, a technique called "differential photometry."

Using more reference stars, and reference stars which are closer in apparent colour to the target permits greater certainty that observed variations are intrinsic to the target object, and not attributable to atmospheric effects or to effects intrinsic to the reference stars.

Small adjustments to the pointing of the telescope, by translation or rotation, can permit the inclusion of more reference stars which more closely match the target. This project is intended to provide a reliable, automated process for making these adjustments, as outlined further in Chapter 4.

3. Computing Background & Concepts

This is a scientific computing project, developed with a Top-Down design philosophy. As such it demands a robust assessment and planning process which identifies the project's needs and compares them with the resources that are available. With a clear understanding of this comparison, it is possible to develop a plan for providing a solution to the computing problem.

The fundamental goal of this project is to develop and test a system to probe the SDSS Catalogue, and large astronomical catalogues in general, seeking stars which are highly suited to differential photometry, and hence optimising the potential detection of exoplanets by the transit method.

Therefore, from a computing perspective, it is imperative to understand SDSS and the demands that the analysis will place upon the computing resources available before beginning the software design process. Section 3.1 assesses the requirements of the project with regards to an analysis of SDSS.

This assessment indicates a need for high-performance computing resources to provide capacity beyond those available from the individual computers available to the project. The solution to this problem is to use multiple computers working together in a parallel computing paradigm. Section 3.2 explains the concept of parallel computing in detail. The spectrum of parallel computing paradigms is examined and the options available to this project considered. The paradigm chosen for this project is grid computing, and the particular implementation of this paradigm used is Grid Ireland. The architecture of Grid Ireland is explained in Subsection 3.2.1

The format in which SDSS data is stored places requirements on the software systems that can be used in the project. Other considerations such as the most suitable programming paradigm for the needs of the project, the level of the programming language chosen, and the relative speed and efficiency of the programs generated are defined and taken into account. The software solutions available and the options selected within each of these parameters are detailed in Section 3.3.

3.1. Assessment of Computing Requirements

This project uses novel techniques to analyse the SDSS catalogue, with the intention of developing a system to data mine large astronomical catalogues on a star-by-star basis to identify those most amenable to study by differential photometry. The exact astronomical parameters used to perform this analysis, and the benchmarks used to assess the performance of this system are discussed in Chapter 4. It is, however, essential to have a basic understanding of SDSS to determine the scale of the project, and thus the scope of the resources needed for this project to be viable.

As discussed in Section 2.3, SDSS DR7 is an astronomical catalogue consisting of 4.27TB of data. This data includes entries on 357,175,411 [5, 6] unique objects. These entries are stored in 421,388 astronomical standard Flexible Image Transport System (FITS) format table files. [5] These files take the extension .fits (or .fit for systems limited to three character extensions.)

FITS files consist of one or more Header and Data Units (HDU), each of which consist of a header and the data that header describes. [86] The FITS format is described in more detail in Subsection 8.3.1, but its use requires that any software used in this project have access to FITSIO (fits Input/Output) Libraries as discussed in Subsection 3.3.2.1.

An experimental estimate of the time taken to fully process a single target star in the SDSS Catalogue using an early prototype of the Application Processing Interface (API – defined in Subsection 7.2.1) on a single PC at ITTD was between 0.25 and 1.0 seconds. Multiplying this by the number of unique celestial objects in SDSS indicated an overall processing time of between 2.8 and 11 years. This indicated a need for a high performance computing system as discussed in Section 3.2. Later refinements to the system led to a reduction in this time as indicated in Section 13.2.

3.2. High Performance Computing

High performance computing is an umbrella term covering a spectrum of different techniques. [87] All take the same fundamental approach to increasing the speed at which a computing challenge can be completed – that of parallel processing. [87] [88] Parallel processing is where many calculations are carried out at the same time on

separate processors. This allows the overall computing problem broken down into multiple smaller tasks which can be carried out concurrently. [89] [87]

The techniques used to divide and solve the computing tasks range from classic supercomputers, where many processors are located in close proximity to one another and connected by a high speed bus [90]; to volunteer computing systems where many widely distributed processors are connected by standard internet connections. [91] A means of distinguishing between different paradigms of parallel computing is the degree of distribution, [89] which typically has an inverse relationship with the speed of connectivity as shown in Figure 3-1. There is considerable overlap between the terms used, as each represents a spectrum of related technologies, indicated in the diagram with shades of grey. [87] [88] [89]

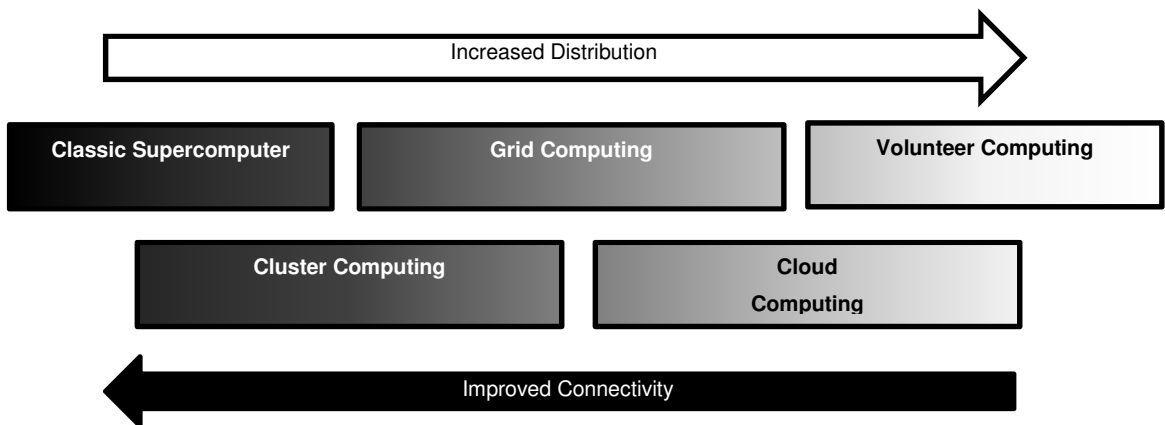


Figure 3-1: Spectrum of parallel computing

Of the five paradigms shown in Figure 3-1, three were unsuitable or unavailable for this project. Classic supercomputers use highly specialised hardware and software which was not available at ITTD. Volunteer Computing projects such as SETI@home or folding@home must, by definition, incorporate recruitment of the volunteers, [91] which was not within the scope of this project.

Cloud computing, which may be considered an evolution of grid computing, [88] was an emerging technology when this project was envisaged, and was not a viable option at the time. Conversion of the project software to the Cloud paradigm is considered in Section 14.2 as a suitable future expansion of the project.

The two remaining options, Cluster and Grid Computing were compared, both on the merits of the paradigm as a whole and in terms of the particular implementations that were available to the SCG at the time of the development of the project.

Cluster computing is where a number of physically proximate computers are connected to one another by means of high speed Local Area Network (LAN) connections. [89] Cluster computing can be differentiated from classic supercomputing by the use of off-the-shelf components as opposed to specialised equipment. [90] In this paradigm, each computer on the cluster, known as a node, runs a separate instance of a standard operating system (OS). A private cluster of 40 computers was available at ITTD under the auspices of the School of Computing. This cluster operated using a Message Passing Interface (MPI) system. MPI is a specification for message passing between individual computers in a cluster situation. [92]

	Grid (Grid Ireland)	Cluster (ITTD School of Computing)
Computers	1152	40
Storage	10TB (allocated)	<1TB
Management Software	EGI/gLite	MPI
Connectivity	1Gbps Ethernet LAN, 10Gbps WAN	1Gbps Ethernet LAN
Control	External (TCD)	Internal (ITTD)

Table 3-1: Comparison between Grid and Cluster solutions. Green indicates the superior option in a category, while red indicates the inferior.

Grid computing uses a number of physically distributed computers connected to one another using a network, which may be public, private or the Internet. Grid computing is similar to cluster computing in that the individual nodes run separate instances of the OS. Grid computing typically incorporates middleware to manage the assignment of elements of the project to the nodes. Grid Ireland was a project to make grid computing available to Irish scientific computing projects. The grid consisted of 1152 processors at the OpCentre in Trinity College Dublin (TCD) and a number of other centres around the country. [93] It was operated using the European Grid Infrastructure (EGI) including the gLite middleware package which provided robust job management systems. [93] [7] Dedicated storage of 10TB was made available to the SCG by Grid Ireland. Table 3-1 shows the comparison between the Grid and Cluster solutions available.

The grid paradigm, implemented by Grid Ireland, was selected for use in this project. As can be seen from Table 3-1, it outperformed the Cluster option in most relevant categories.

3.2.1. Grid Ireland

Physically, Grid Ireland consisted of 1152 computers, also referred to as nodes or worker nodes. [93] Each node was an off-the-shelf PC running a separate instance of Scientific Linux version 4.5. This means that each has its own local Unix File System (UFS.) These included 768 computers at the OpCentre at TCD and up to 384 computers located at eleven locations around the country, of which 50 were at ITTD.

Note that due to Grid Ireland's intentionally heterogenous structure, nodes were of different physical configurations, including RAM, local HDD and CPU capability. The user could specify requirements for these parameters as part of a grid job submission as discussed in Subsection 7.2.4. [94]

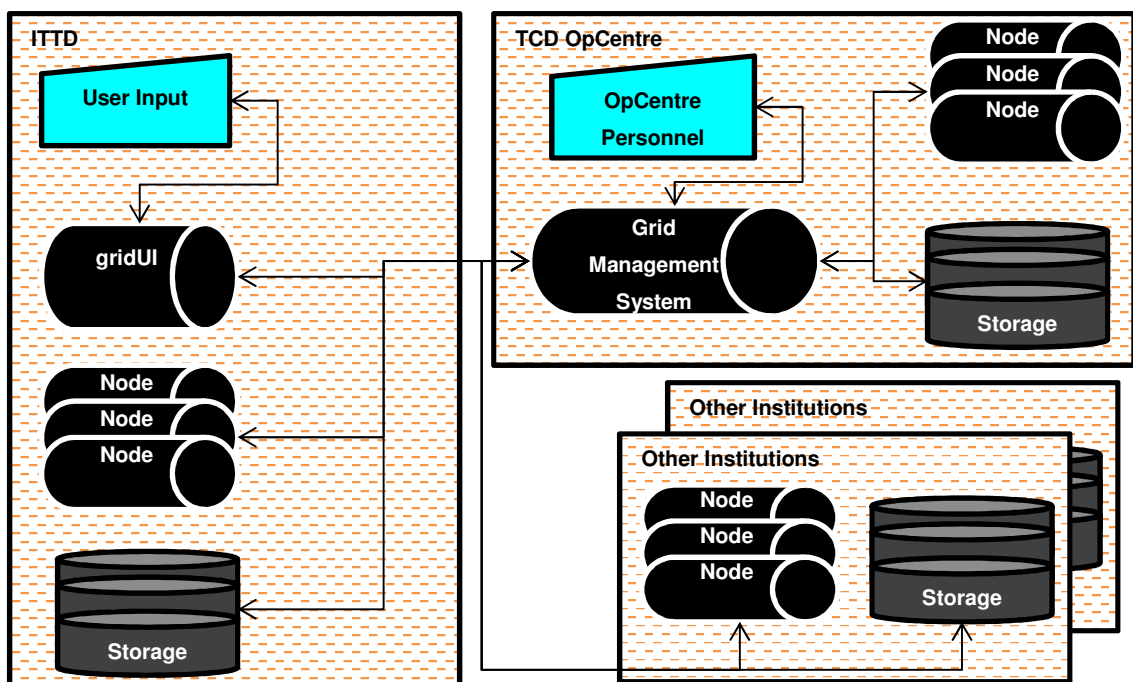


Figure 3-2: Schematic of the physical organisation of Grid Ireland

These computers were connected locally by 1Gbps Ethernet LAN connections and the locations were connected to the OpCentre's 10Gbps network by means of 10Gbps dedicated line network connections. [93] Over 700TB of data storage, also distributed

nationally and managed through the OpCentre was available to the grid, of which 10TB was allocated to the SCG. [93] In general, data access was limited to local node data, or data accessed through the GMS, such as the Logical File Catalogue (LFC.)

Access to the grid was provided by means of gridUI (grid user interface), a dedicated computer which acted as a gateway to the grid, located at ITTD. This computer used the same operating system and shell, and had the same access to the grid as a grid node and as such was ideal for testing and debugging grid software. It was accessed by the user through the secure shell (SSH) remote login routine. Figure 3-2 shows this physical organisation in schematic form.

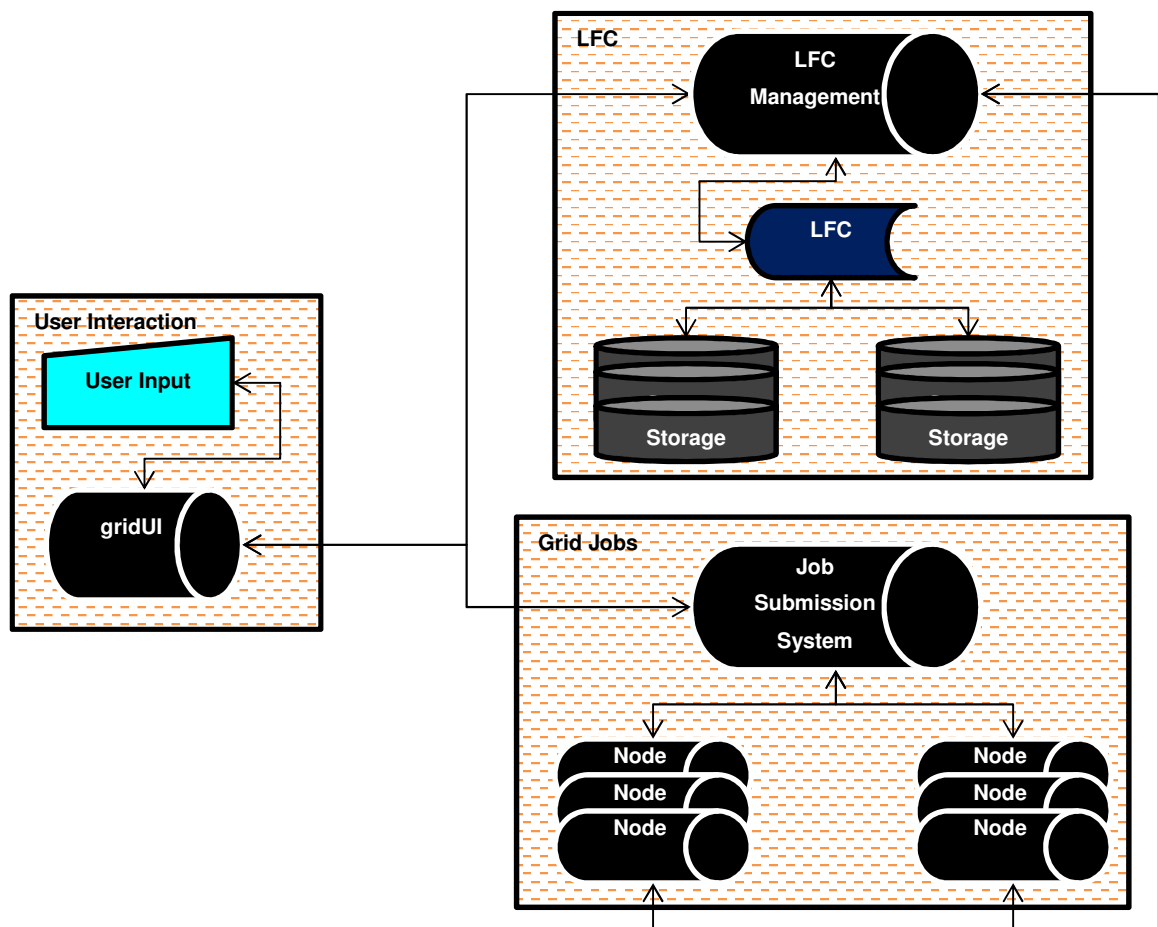


Figure 3-3: Conceptual structure of the grid from the user perspective.

Grid elements only interact with one another thorough the Grid Management System (GMS.) [93] [7] The GMS has two major components: the Logical File Catalogue (LFC) and the Job Submission System (JSS). [7] These two elements are designed in

such a way the physical location of a computing resource is controlled automatically by the GMS. [7] As a result, design choices are shaped more by the conceptual design shown in Figure 3-3 rather than the physical layout shown in Figure 3-2.

The LFC system consists of three elements.

- The first is a catalogue of Logical File Names (LFN) linked to their corresponding Globally Unique Identifiers (GUID) and Storage Universal Resource Locator (SURL). [95] LFNs are designed to be human-readable and are structured to mimic a standard UFS. GUIDs are a unique string of the form `guid:<unique_string>` and are used as a primary key for each file in the catalogue. SURLs refer to the location where the file is physically stored.
- The physical storage referred to by the SURL is the second element of the LFC suite. [95] [7] Unless the user specifies otherwise, the physical location of files is assigned automatically by the GMS.
- Data in the LFC cannot be directly accessed, but rather must be accessed by means of the final element of the LFC – the set of `gLite` middleware commands used to interact with the data in the LFC. [7] These commands are designed to mimic the UNIX shell commands. For example, the `gLite` command `lfc-ls` will list the contents of a directory specified by its LFN in the same way that the `ls` UNIX command will list the contents of a directory in the UFS. [7] [96]

Similarly, the user does not interact with the grid nodes directly. Work is instead assigned to the nodes by the JSS. [7]

Users submit individual tasks to the JSS by means of grid jobs. Grid jobs are each defined by a single file written in the Job Description Language (JDL/.`jdl`). JDL is described in more detail in Subsection 8.3.6, but fundamentally consist of a series of name-value pairs which fully define the task assigned to the grid and specify any requirements that that task may have.

A JDL file must include the path to an executable file which will be copied to and executed on the worker node (WN.) [7] Additional requirements may include data files

that are required for the job. [7] Unless the user specifies otherwise in the job, the physical location of the worker node is selected automatically by the GMS.

Jobs are submitted to the GMS using `gLite` commands. When these commands are executed, the JDL file is sent to the GMS, and the executable file, together with any data files specified in the job, are uploaded to the grid. [7] These files are placed, together, in the working directory of the WN to which the job is assigned. This is an essential consideration for the design of the executable software.

Further `gLite` commands exist for the monitoring and management of grid jobs. [7] [95] Grid jobs, once submitted to the scheduler are given a status to indicate their progress from “submitted” to “cleared” which may be monitored by the user manually, or automatically as used in this project and discussed in Subsection 7.2.4. [7]

Grid jobs may fail to complete for a variety of reasons, including grid problems, such as node crashes or software problems such as memory leaks or missing data. The GMS does not automatically resubmit failed jobs, instead requiring that the user monitor the progress of jobs, and resubmit them if appropriate. Checkpointing may be implemented by the user using the LFC and suitable scripting techniques if desired, but is not part of the default system. [7]

3.3. Software Specifications

As shown above, external factors such as the format data is stored in place constraints on software options and project objectives set requirements.

Several core concepts of computer programming are also relevant to the choices that must be made for this project, and are described in detail in Subsection 3.3.1. These concepts complete the framework for the choices to be made in Subsection 3.3.2. For each, the options available are explained in detail. Their advantages and disadvantages are compared to provide a rationale for the choice that was made.

3.3.1. Computer Programming Concepts

Providing a software solution to a problem means choosing a programming language in which to write that solution. The most fundamental requirement of the chosen language

is that it be compatible with the computing problem posed. This compatibility can be assessed on two criteria.

Firstly, there must be a suitable implementation of that language on the OS to be used. This implementation takes the form of a system to translate the program the programmer has written into machine code. [97] The mechanisms for this translation are discussed further in Subsection 3.3.1.2.

Secondly, in any programming language, there are collections of instructions that invoke particular behaviour in the computer. These collections are called Libraries. For a programming language to be considered compatible with a problem there must be suitable instructions available to the programming language to access any data that is needed. These instructions may be part of the standard library of instructions built into the language or may be available through a custom library which must be linked to the program by the programmer.

Within this project, where there are multiple languages which meet these criteria, three further concepts of computer programming are considered when choosing among those options. Those concepts are the abstraction level of the available languages, the translation mechanisms available, and the programming paradigm for which the language is designed. These concepts are explained below.

3.3.1.1 Language Levels

The level of a programming language is a description of the degree of abstraction from the processor's instruction set, also known as machine code. The instruction set is a set of patterns of bits, known as instructions, which make up the fundamental building blocks of software. These instructions correspond to the physical design of the processor, and cause it to carry out their respective functions.

Abstraction is a means of enabling the human mind to comprehend a system by understanding *what* its subsystems do without knowing *how* those subsystems operate do it. [98] For example, assembly language replaces the bit patterns of the instruction set with mnemonics which are easier for the programmer to read. This corresponds to a single layer of abstraction.

A programming language is said to be higher level than another if it has more levels of abstraction, and has more automated features, however different authors use the terms “High Level” and “Low Level” differently. As an example, C may be considered a high level language (as per Deitel & Deitel, 2001 [99]) because it uses natural language elements and allows for features such as loops. On the other hand, C may be considered a relatively low-level language (as per Kernighan & Ritchie, 1988 [100]) because it allows for low-level access to data. Because of the combination of features it could be described as mid-level, as per Schildt, 1987. [101] This Thesis, therefore, refers to the level of a language only relative to others to which it is compared.

3.3.1.2 Software Translation

Unless software is written in machine code, it must be translated into machine code in order for the computer to be able to run it. [97] Programs written in Assembly Language usually require a one-for-one translation of the assembly mnemonics into machine code, using a program called an assembler. [102] Programs written in higher level languages may be compiled or interpreted. These processes are carried out by programs called compilers or interpreters, each of which is specific to the machine and source languages to be translated. [103]

Compilers translate source code into machine code before the program is executed. [104] [97] This machine code can then be executed at a later date, and may be executed repeatedly. Compilation has the advantage that it is much faster to execute at run time, because the translation has already taken place. Its chief disadvantage is that a program must be complete before individual functions can be tested. [97] [103]

Interpreters read, translate and execute a program one statement at a time at run time. [97] [102] This is a slightly inefficient way of executing programs, as each statement must be translated each time it is encountered, especially if there are loops in the program. [102] Languages such as Python and shell scripting languages are examples of interpreted languages. [105] Virtual Machines, (VM) such as the Java Virtual Machine, are another class of interpreter. The advantages of using interpreters is that functions can be tested as they are designed, and, in the case of VM, that changes made to a language specification can be made to a single compiler, rather than having to update a family of compilers. [103]

Languages are often described as compiled or interpreted. This description is based upon how they are designed to be used, but this categorisation is not absolute. It is possible to create an interpreter for a compiled language, or vice versa. As an example, Ch is an interpreter for C, a compiled language. [106] In this Thesis, however, languages are described according to their designed use, and were used in this manner in the course of the project.

3.3.1.3 Software Paradigms

Burdett et al. define two main categories of programming language, Imperative and Declarative. [102] Within each of those Burdett et al. include two paradigms as shown in Figure 3-4. Procedural and Object Orientated languages are subsets of Imperative Languages. [102] [107] Functional and Logic programming are subsets of Declarative programming. [102] Many languages have features of more than one paradigm of computing. C++ is an example of a hybrid language: it supports both object-oriented and procedural programming; and a given C++ program may contain elements of both paradigms [108].

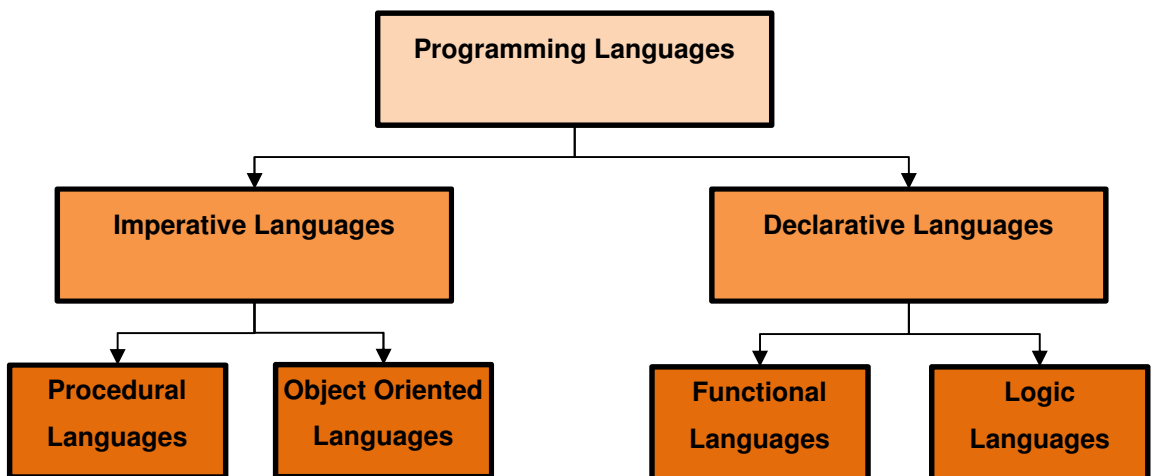


Figure 3-4: Programming Language Paradigms

Imperative languages are languages in which the programmer gives the step-by-step instructions to the computer to complete the program. These instructions include the order in which the instructions are to be carried out. Commands are usually executed in the order they appear unless a branching statement interrupts this model. [103] [102]

Procedural languages, such as C, are sometimes referred to as Imperative languages. [107] In this Thesis the distinction is made that procedural languages are a subset of imperative languages in which program statements are grouped into subroutines, also called functions or procedures. [102] [107] Procedures may be considered additional levels of abstraction, created by the programmer above those built into the program, which are referred to as “primitive statements.” This procedural abstraction lends itself to a process of stepwise refinement, a key element of Top-Down design as discussed in Chapter 5.

Object-Oriented languages, such as C++, are imperative languages which include the concept of objects. Objects contain both program routines (called methods) and the data being processed (called properties.) [102] Object orientation may be considered an additional level of abstraction above that available to procedural languages. [103] Software written in an object oriented manner is considered more reliable and more reusable because self-contained objects are easier to control. [107] [102] While object-oriented languages can themselves be considered a subset of procedural languages, [102] languages described as procedural in this Thesis should be understood to exclude object orientation unless otherwise specified.

Declarative languages, including SQL, consist of statements that specify the properties of the results of the program, in contrast to imperative languages which describe the means by which those results are generated. [102] These statements are used in conjunction with a database or a set of rules to identify results that match the query

Functional languages, such as Wolfram Language – the underlying language of Mathematica, are a subset of declarative languages wherein computation is expressed in the form of mathematical functions. [102] Purely functional languages do not include elements of program state, such as variables. This eliminates side effects that can occur in procedural languages – where a change to a variable can cause a function to provide different results. Instead, in Functional Programming, calling the same function with the same arguments will produce the same results every time. [109]

Logic Programming languages, such as ProLog, are based on the principle of formal logic. Logic programming works by means of a series of statements of facts and

definitions of inference rules. [110] These rules are used in a form of automated reasoning to provide the answer to a query. [102]

3.3.2. Software Solutions

Four challenges exist within this project that require a software solution. The first is accessing the SDSS data and retrieving it from its online archive. Once that data is available for the project, the second software requirement is to analyse and manipulate that data in the form of FITS files to produce the astronomical results. Next, a solution must be developed to access and use the grid. This may be considered under two headings – accessing the LFC and managing grid jobs. Finally, as the three prior requirements are likely to have divergent solutions, a system must be developed to connect them together and provide a system to control the workflow and make the results available to the end user.

3.3.2.1 Accessing SDSS

SDSS provides two principal mechanisms for accessing SDSS. Those are the Catalogue Archive Server (CAS) and the Data Archive Server (DAS.) [111] In addition to these systems, a number of third-party systems exist including, for example, “Tool for OPERations on Catalogues And Tables” (TOPCAT) which uses Virtual Observatory (VO) standards to provide access to multiple catalogues including SDSS. [112]

The VO is not a single technology, but rather an approach and a set of standards which are intended to provide uniform access to multiple astronomical data sources. The International Virtual Observatory Alliance (IVOA) is a collaboration of many astronomical data projects (both users and generators) which determines these standards. [113]

One such standard promulgated by the IVOA is the Astronomical Data Query Language (ADQL.) ADQL is derived from SQL and is designed to enable queries which are specific to astronomy to be performed on the many tabular datasets and the relational databases which form the VO. [114] Output from ADQL queries is recommended to be delivered in the form of VOTables [115], an XML format based on the FITS binary table format which is built to be particularly suitable for astronomical query output as delivered by the VO. [116]

The use of the Python programming language, which is built with support for a large suite of libraries [105] and with a design which enables it to be used as a “glue language,” linking together components of a software system written in many languages [117] enables the user to incorporate ADQL queries to VO enabled systems such as SDSS by using the `astropy` library. [118] `astropy` is a suite of astronomical software designed to enable Python programs to access most core tools used in astronomy and astrophysics. [118]

The use of VO-derived systems, for the most part, requires access to VO web services at runtime. [118] However, as discussed in Subsection, 3.2.1, the nodes at Grid Ireland are isolated from general internet access, and can only interact with data provided via the GMS. This makes the VO approach unsuited for work with the HPC solution used in this project. Instead, the work of this project focussed exclusively on the use of the SDSS provided tools: CAS and DAS as discussed below.

The CAS is an SQL Database with a web interface which permits fast searches with time and row limits using the SkyServer tool [119] and larger batch queries using the CasJobs site. [6] The DAS is a collection of FITS images and tables containing the outputs of the SDSS observation pipelines. [111]

Data accessed through the CAS is provided to the user in the form of Comma Separated Value (CSV), HyperText Markup Language (HTML) or eXtensible Markup Language (XML) files. Of these, CSV files were used in this project when accessing the CAS as it is the most light-weight. [120] Queries submitted via SkyServer are fast, but limited; while queries submitted via the second method called CasJobs – a batch processing mechanism – was nevertheless found to be unpredictable in response time. [6] Results for similar queries submitted on several occasions delivered response times that varied from 10 minutes to 1 day. This unpredictability makes the CAS unsuitable for interactive operations at run time. Repeat access to data via CAS means repeatedly downloading the data. Access to data via the CAS requires a fast, reliable Internet connection from the processing machine to SDSS, which cannot be guaranteed.

The SDSS data needed for this project is called the SDSS Calibrated Object lists and is stored in the DAS as `tsObj*.fit` files. [121] These files contain FITS tables, which

are discussed in more detail in Subsection 8.3.1, store numerical data as floating point numbers, which are much more compact than their ASCII representation.

These files in the DAS are stored in a directory and filename tree that is defined by four observational parameters: `run`, `rerun`, `camcol` and `field`. This directory structure is fully explained in Subsection 8.2.5.1. The correspondence between the observational parameters and the astrometric coordinates of the objects observed cannot be calculated at run time. Instead, they must be identified by means of a catalogue query through the CAS. [122] The mandates a series of SQL queries to identify these fields, and shell scripts to bring these files together at run time as discussed in Subsections 7.2.4 and 7.2.5.

These files can be downloaded in a non-interactive mode by means of the `wget` application (or any other bulk download utility) and stored locally or on a grid resource such as the LFC. [122] `wget` recreates the directory structure used in the website it downloads from, which preserves the correspondence between file name and astrometric location. [123] This permits repeat access to the data without a need for repeat downloads of the data. LFC response times for access to the LFC were experimentally assessed to vary between 1 and 5 seconds.

	CAS/Online Access	DAS/Download
Data Format	SQL Database (CSV output)	FITS Tables
Storage	Online	Local (Grid)
Interaction	Integrated	Separate (via CAS)
Response Time	Unpredictable (<10 mins)	Predictable (<1s)
Internet Access Frequency	Download many times	Download once
Data Access	Web Interface	LFC
Repeat Access	Same speed each time	Faster after first time

Table 3-2: Comparison of SDSS Data Access Options. Green highlighting indicates preferred option.

As is shown in Table 3-2, downloading data via the DAS has many advantages over use of the CAS at runtime. As a result, this is the chosen method for interaction with SDSS in this project. Limited use of the CAS is still required to map the FITS file paths to astrometric coordinates. SQL queries must be written to provide this information. These are used as a pre-run step to guide grid jobs, resulting in no run-time delays during job execution. SQL is considered a declarative language, although it includes

limited imperative elements. [124] This paradigm guides the design of SQL queries as shown in Subsection 7.2.5.

3.3.2.2 Processing FITS Files

Accessing FITS data requires use of a language with suitable FITSIO Libraries. Five options are considered here: C, C++, FORTRAN, Python and Java.

C is a general purpose programming language designed for the UNIX operating system. [100] It is “closer to the machine” than FORTRAN or the other language options available, which makes it a better choice for dealing with system functionality such as I/O. [125] Access to FITS files is provided by means of the CFITSIO Library, which is the basis for the other FITSIO Libraries which are available. [126] [127] [128]

C++, originally referred to as “C with classes,” is an object-oriented language derived from C. [108] Software written in this paradigm is considered to be easier to maintain than procedural software like C. [125] CCFITS is an object-oriented interface to the CFITSIO Library for use with C++. [127] CCFITS was subject to known bugs at the start of this project. [127]

FORTRAN is an early high-level language developed primarily for mathematicians and scientists. [125] It is designed such that its basic components resemble those in a mathematical formula. FORTRAN is considered the fastest of the languages that are available for numerical computing, but there can be a mismatch between the variables and their representation on the computer which can cause Input/Output (I/O) inefficiencies. [125] A FORTRAN interface is available for CFITSIO, which enables use of this language with FITS files. [129]

Python is an interpreted, object oriented language which is designed to provide “a simple, easy to learn syntax” [105] and a large suite of libraries to allow for specific operational requirements and to provide an interface with existing systems, notably PyFITS, first released in 2002, which provides an interface to FITS files. [130] At time of writing, PyFits had been incorporated into AstroPy. [130] [118] Python programs take longer to run than the other languages discussed here, but are easier to write and understand. [105] [131] To provide this ease of use, Python is designed as a higher level language than any of the others discussed here, and is itself implemented in C. [105]

Java programs look similar to C and C++ programs, and they use the same programming Paradigm as C++. However, Java code is often interpreted at run-time, which means it tends to be slower than the other options considered here. [125] The JFITSIO Library is a java library wrapping the CFITSIO Library to allow Java Native Interfaces (JNI) access to FITS Files. This library was not a mature technology at the start of this project, development on JFITSIO having started in May 2007. [128]

	C	C++	FORTTRAN	Python	Java
Paradigm	Procedural	Object-Oriented	Procedural	Object-Oriented	Object-Oriented
Level ^[a] [100] [108] [125]	1	3	2	5	4
Library [127] [128] [129]	CFITSIO	CCFITS	CFITSIO ^[b]	PyFITS/ Astropy	JFITSIO
Translation	Compiled	Compiled	Compiled	Interpreted	Interpreted
Numerical Computation Speed ^{[a][c]} [125] [132] [131]	4	3	5	1	2
Expertise Available	Yes	Yes	No	No	Yes
Notes: [a] ranked order lowest to highest. [b] CFITSIO with FORTRAN Interface [129] [c] interpersonal difference between programmers can dominate over differences between programming languages [133]					

Table 3-3: Comparison of programming languages with FITSIO Capability

Of the five options available, C was chosen as the primary programming language. The FITSIO Libraries for each of the other three languages are wrappers for the CFITSIO Library, indicating performance at best equal to that for CFITSIO. Its relatively low-level system functionality provides an I/O performance advantage over the higher level languages. While FORTRAN is slightly faster computationally, the performance is, as Ashby says, “so close as to make any distinction between the languages on performance grounds specious.” [125] Expertise within the SCG included C, C++ and Java, but not FORTRAN or Python at time of start of the project. A side-by-side comparison between the five available programming languages is given in Table 3-3.

With the decision to use C, the programming paradigm to be used in this aspect of the project is therefore procedural, and the design of software in Chapter 7 must be

reflective of this. This decision also affects the choice of Integrated Development Environment (IDE) discussed in Chapter 9.

3.3.2.3 Interacting with Grid

Grid Operations are not identical to operations in a classic, serial computing model. The differences introduced below, and examined in detail in Section 7.1 impose a series of constraints on the design of the software solution to the research problem.

As discussed in Subsection 3.2.1, access to grid resources such as the LFC requires the use of externally provided software, that is, the `gLite` middleware package. [7] Submission of jobs to the JSS is not a matter of typing a simple executable command. Rather, grid jobs – files specifying the parameters of a grid job which are written in JDL – must be submitted using further `gLite` commands. Each of these aspects of the use of the grid places further design constraints on the project

- **LFC Design Constraints**

Access to the LFC requires authentication every time a file is copied to or from it. The time taken for this authentication is experimentally estimated to be approximately two seconds, in addition to the time taken to transfer the file over the network. This sets a design parameter whereby the number of LFC access commands must be minimised where possible.

- **JSS Design Constraints**

Grid job submission requires one JDL file per grid job to be executed on a node. [7] Since the project is designed to require many such jobs, it is necessary to develop a system to generate, store, submit, monitor and retrieve these jobs. This system must create many similar files with different parameters to complete the analysis.

In addition, the size of grid jobs must be selected in order to maximise efficiency. Initial experimental estimates indicate that the submission of a grid job takes approximately ten seconds, and the interval between job submission and the worker node beginning the execution of its task is approximately five minutes, regardless of the size of the job. These intervals may be considered dead time. Therefore, the designer must maximise the length of the grid job to minimise the fraction of time thus wasted.

However, grid jobs in Grid Ireland are also not permitted to run for longer than three days. [134]

The management of these design constraints is discussed in Section 7.1. Section 9.2 explains the practical steps taken to implement this design. Finally, Section 13.3 provides a series of metrics on the success of this implementation.

3.3.2.4 Shell Scripting

A shell is a Command Line Interpreter (CLI) which executes commands which are input as text either manually or in the form of shell scripts. Shell scripts are programs that are run directly by the shell. They consist of a series of commands in the same syntax as is used when submitting instructions at the command line, and may include more sophisticated features common to programming languages such as control structures and variables. [135]

The use of these programming features enables the programmer, for example, to repeat the execution of a command until a condition is met, or to run the same program with a series of different arguments. [135]

UNIX operating systems typically include a number of different shells, each of which has its own syntax, and shell scripts written in one shell may not be compatible with another. There are too many options for shell scripts to list and compare them all here. [135] The Bourne-again Shell (Bash) is the shell that is used to tie together the various elements of this project. Bash was chosen for the following reasons:

- Bash combines the best features of the most popular shells, the Bourne shell (sh), the C shell (csh), and the Korn shell (ksh), [135] including
 - Control structures
 - Piping
 - Wildcards
 - Ability to access and edit text files
- Bash is the default shell on Grid Ireland Computers [134]
- Bash is one of the most widely used and supported shells

- Bash scripting behaves similarly to the imperative paradigm of programming used elsewhere in this project

This project uses three disparate software elements as described above. In order to be used in the grid, these must be linked in two areas: management and execution. Both links are achieved by means of bash shell scripts

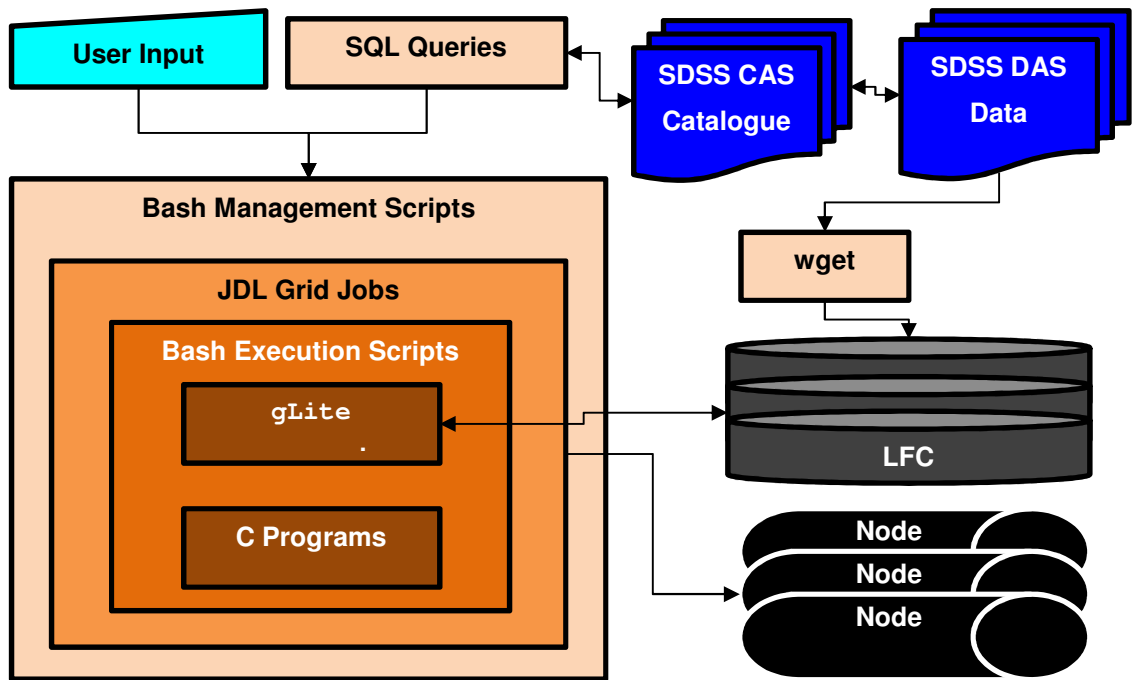


Figure 3-5: Project Software Strategy

As shown in Figure 3-5, the bash management scripts take as input the results of SQL queries to the SDSS CAS. These inputs are used to generate grid jobs, together with input from the user specifying job size and with other parameters. Other management scripts submit the jobs thus generated.

Each grid job may only specify a single executable. [7] This means that Bash scripts must be developed to bring together the various elements that are required for a grid job. Those may include `gLite` commands to download data from the LFC to the node, programs written in C to analyse the data, and `gLite` commands to return output files to the LFC.

Figure 3-5 also includes an illustration of the mechanism by which the SDSS data is accessed: data from the DAS is downloaded to the LFC using `wget`, which preserves

the directory structure and file naming convention of the DAS. Grid jobs are assigned elements of this data to access by the management scripts. Management scripts take input from SQL queries of the CAS. The CAS includes information on the correspondence between the observational parameters used to name the files and directories in the DAS and the astrometric position of the objects contained in those files.

3.4. Conclusion

The objective of this project is to use novel techniques and powerful computing resources to develop a system capable of analysing the SDSS Catalogue. The output of this analysis is a catalogue of stars which are amenable to observation by differential photometry and thus suitable for observation for Exoplanets by the Transit Method.

As demonstrated in Section 3.1, this analysis requires computing power beyond that which is available from a single over-the-counter PC. The analysis also requires a robust software solution to access and process SDSS data.

To provide the computing power, the project examines several options for High Performance Computing as discussed in Section 3.2. The solution chosen, based on the available options, is to use Grid Ireland. Subsection 3.2.1 provides a detailed description of the physical and conceptual structure of Grid Ireland. This structure itself sets boundaries and requirements on the software developed for this project.

The software solutions to the various challenges posed in this project can be categorised in four areas: SQL, C, grid and Bash. SQL queries provide the information used by Bash scripts to manage grid jobs. Grid jobs execute further bash scripts, which use grid middleware to access the data, and C Programs to process that data to produce astronomical results.

It within this framework that the detail of the project design, fully explored in Chapter 7 is developed.

4. Project Objectives

The objectives of the project are divided between astronomical goals, and computing goals. The astronomical goals set out in Section 4.1 are based on the requirements set out in Chapter 2 for improved differential photometric precision. The computing goals defined in Section 4.2 are to demonstrate the use of grid computing for a large-scale astronomical data mining project.

4.1. Astronomical Goals

The scientific objective of this project is to provide an automated system to improve differential photometry precision by providing more and better reference stars to observers. Two scenarios are envisaged for the use of this system.

In the first, consider an observer who wishes to observe a specific target. As demonstrated in Subsection 2.2.3.5.3, the ideal point of aim for their telescope may not be directly at the target but rather offset from the target. This offset can be adjusted to maximise the quantity and quality of available reference stars. Subsection 4.1.1 lays out the requirements of this project to automatically determine the point of aim for a given target.

The second scenario relates to the creation of catalogues of objects, each with an appropriate pointing to maximise the precision of differential photometry available, and with a figure of merit describing the quality of said photometry. An observer who wishes to observe objects which meet certain criteria with great precision, but who does not have an a priori preference of which particular object to observe may thus choose the objects for which the best reference stars, and thus the best precision, is available. Subsection 4.1.2 describes the project requirements to accomplish this goal.

Finally, the outputs of this project, including the catalogues, must be analysed. This project is intended to develop and demonstrate a capability. This capability must be analysed as to the nature, quality and limitations of its outputs. An understanding of this analysis is required before future work can be proposed to make use of and expand upon this capability. The analysis of outputs required for this project is discussed in Subsection 4.1.3.

4.1.1. Individual Field Optimisation

Choosing the optimum point of aim for a given target requires several key factors. The primary requirement is that the point of aim be positioned such that the target is included in the field of view.

The field of view must also be positioned in such a way as to include a number of reference stars. Viable reference stars in the vicinity of the target must be identified. Unless all of these reference stars can be included in the same field, a choice must be made between which ones to include, and which to exclude.

This process must be constructed in such a way as to lend itself to automation. An algorithm consists of a sequence of steps to solve a problem. Therefore the optimisation of a field needs to be broken down into a series of logical steps which can be programmed.

When defining the algorithmic solution to the problem, practical limitations may demand that certain constraints be placed on the solution. For example, rotating the field of view greatly increases the computational complexity of the solution. Therefore a pragmatic solution – which only considers the translation of the field of view rather than permitting translation and rotation – would be more tractable in the first instance.

A naïve solution to the problem of choosing between possible pointings would be simply to count the number of viable reference stars in the field at that pointing. However, Young et al. [51] and Milone and Pel [1] suggest that close match between target and reference star on colour is important for precision differential photometry. Since a perfect match is unlikely, a sliding scale which calculates a *rating* for each reference star should produce better results if appropriately calibrated. The rating system developed for this project is discussed in Subsection 5.3.1. A given field can have a *score* calculated for it by an appropriate combination of those ratings. The mechanism used to combine scores into ratings is discussed in Subsection 5.3.2. The optimum pointing is thus the pointing with the highest *score*.

Different observational conditions may provide different criteria for observation, such as available dynamic range, and capacity for resolution, which will impact the choice of

viable reference stars. Therefore a system developed to determine these must allow for flexible input parameters to control these criteria.

Since each combination of telescope and camera have their own size and shape of field of view, a pointing developed for one telescope will not be optimised for another. Therefore any system developed to automate this process must allow for the user to input parameters designating the field of view size amongst the parameters discussed above.

The first goal of this project, therefore, must be to create a system to optimise the pointing for a given target. Such a system must be able to

- identify potential reference stars
- provide a *rating* for each
- determine potential pointings for a given telescope
- identify which pointing would provide the highest *score* for that target
- do so in a computationally efficient manner
- allow modular customisation of the scoring system
- provide a user input mechanism to allow for a variety of parameters to the algorithm

4.1.2. Production of Catalogue of Optimised Fields

If it is possible to determine an optimised field for a single star, it follows that repetition of this process for multiple stars would produce a series of pointings, one for each star. Collation of these results can produce a catalogue which can be searched and sorted on a variety of criteria.

Using a consistent scoring system across an entire catalogue analysis allows the end user to compare one target with another. If the catalogue is then sorted by this score, the "best" and "worst" targets in the catalogue can be identified, and follow-on observations can emphasise targets with higher scores.

A catalogue produced in this manner will necessarily be highly specific in its input configuration, in that a distinct set of observational and telescope parameters must be submitted to the process which optimises the field for each target. Unless these

parameters are the same for all targets in a catalogue, comparison between targets is meaningless. Therefore different equipment and observational conditions require different input parameters. This means that the system of generating optimised catalogues within this project must be developed such that the system can be used repeatedly with different parameters to allow for different conditions.

Because the number of combinations of observational requirements and telescope parameters in the world is very large, it is impractical to attempt to generate suitable catalogues for all of them. Rather, within this project, a more pragmatic approach is taken. The goal of this project is to demonstrate a reliable, repeatable and scalable system which can produce catalogues for a variety of purposes. To this end, two catalogues were to be generated to demonstrate the operation of the system in two different modes, and to provide an initial resource for the astronomical community. These were the Quasar Catalogue and the Exoplanet Catalogue, discussed below.

These two catalogues were generated using parameters for FoV that were towards the low end of common FoV size [136], 10 arcminute for the Quasar Catalogue and 15 arcminute for the Exoplanet Catalogue. [23] These sizes were chosen to allow the use of the catalogues and their pointings with multiple telescopes, including the ones at BCO and Raheny Observatory, because it is possible to use a pointing with a telescope with a FoV larger than the one for which the pointing was developed, but not for one smaller than it, as discussed in Subsection 14.1.3. Similarly, the choice of ± 2 magnitude limit means a dynamic range between brightest and faintest reference of $2.512^4 = 39.82$. This makes it possible to calibrate integration times such that the brightest references do not saturate, while the fainter ones still achieve viable SNRs even with modest equipment.

The second goal of this project is the production of a system to create catalogues of optimised pointings for differential photometry. Such a system must meet the following criteria

- identify optimal pointing for a large number of stars
- produce scores for each of these pointings to permit comparison between them
- accept variable inputs to permit generation of multiple catalogues for multiple criteria

- be scalable to accept large and small target sets

4.1.2.1 Quasar Catalogue

The first catalogue to be generated in the course of this project was called the "Quasar Catalogue." As discussed in Subsection 2.1.3, quasars are variable objects which can be observed using similar techniques to those used to observe exoplanet transits. The generation of pointings for a set of quasars therefore would use similar parameters to those used for exoplanets. In addition, the production of a sortable catalogue of pointings for differential photometry on a large number of quasars has considerable value in its own right. It is intended that these pointings be used in a forthcoming observation project at BCO.

The number of quasars in SDSS is small compared to the number of stars (77,429 for the fourth SDSS Quasar Catalogue [137] vs 357,175,411 [5, 6] for SDSS DR7, used in this project) but still large enough to demand significant computing power to generate a suitable catalogue. Therefore it was possible to use the quasar data set to act as a test for the system to generate catalogues.

In addition, the quasar catalogue provides a means of testing the use of grid computing resources. The quasar catalogue demanded the use of job creation, submission and retrieval routines. These routines can be adapted and reused for subsequent catalogues, including the exoplanet catalogue. Further, the quasar catalogue would provide early metrics on the performance of the grid. (See Subsection 4.2.2 for details on metrics required.)

The purpose of the quasar catalogue is as follows

- produce pointings for all 77,429 quasars in the fourth SDSS Quasar Catalogue
- demonstrate the system for creation of smaller catalogues
- test the grid computing techniques
- create methods for determining grid metrics
- provide preliminary metrics on grid software

4.1.2.2 Exoplanet Catalogue

The generation of a catalogue of candidate hosts for exoplanets, referred to as the Exoplanet Catalogue, constitutes the *Primary* goal of this project. The objective here is to identify pointings for every single star in SDSS by taking the entire catalogue of stars as its target list. In the Exoplanet Catalogue, as discussed in Section 11.4 and Chapter 12, each target has an optimised field generated for it by the algorithm and a score assigned to that pointing.

The production of a catalogue this large (there are 357,175,411 entries in SDSS [5, 6], of which some number, unknown at the start of the project, were stars) demonstrates the ability of the software to scale up from the quasar problem, and highlights and addresses the difficulties in running a data-rich, process-poor project on a grid. The metrics developed for use on the quasar catalogue can be used here to assess the performance of the Exoplanet job.

The purpose of the Exoplanet Catalogue is as follows

- produce pointings for all stars amongst the 357,175,411 entries in SDSS DR7 [5, 6]
- demonstrate the system for generating large catalogues
- provide metrics on the use of a grid in a large-scale, data-rich, process-poor problem

4.1.3. Meta-Analysis

The data produced in the various catalogues created as part of this Thesis takes the form of a large collection of files. In the case of the Exoplanet catalogue, for example, the data within those files amounts to a table of up to 357,175,411 rows and 10 columns. This data may be searched and sorted for relevant outputs for specific purposes as needed by a user.

However, it is necessary to analyse the output data, to allow understanding of the meaning of values such as the core, and to guide further research based on this data. This analysis of already analysed data is referred to in this Thesis as “meta-analysis.”

Close examination of the data permits the characterisation of the dataset, and helps to identify interesting and scientifically relevant patterns and trends in the output data.

It is necessary to take a pragmatic approach to meta-analysis based on software and hardware limitations as discussed in Chapter 12. This includes performing the meta-analysis on subsets of the output data, rather than the entire dataset. These subsets are chosen to be statistically significant and, in so far as practical, chosen to be an unbiased representative sample of the entire data set; while limitations of these subsets are clearly identified, and any conclusions drawn are made with suitable caveats.

The output catalogues, as discussed in Subsection 8.3.1.3, include position and magnitude information (from which colour can be calculated) about the targets and position and score information about the pointings generated. Of these, three variables are suitable for meta-analysis as discussed in Chapter 12: magnitude, colour and score. In addition, it is possible for entries to “fail” as discussed in Subsection 5.2.6 . The meta-analysis must therefore provide the following

- determination of whether any significant distribution exists in magnitude, colour and score, and if such patterns exist, provide a characterisation of those trends
- characterisation of the relationship between magnitude and score
- characterisation of the relationship between colour and score
- interpretation of these relationships, and guidance, as appropriate, for future work
- identification of any failures in the output data, and an assessment of any patterns in failed targets with respect to colour and magnitude

These data can be provided in the form of histograms which count the number of times a variable takes a value that falls in a particular bin. When plotting histograms, the data bins into which data is partitioned must be carefully selected. If the bins are too large, detailed trends in the data may be obscured, but if they are too small, statistical noise effects can come to dominate over the real pattern.

It is also relevant to identify the distribution of descriptive statistics of values for score in a number of ranges of colour and magnitude. This is intended to provide a quantitative assessment of the ranges of colour and magnitude in which unusually high

or unusually low scores are to be found, and provide a sense of where the score for a particular pointing could be considered to fall within the range of values for similar stars. (e.g. is the score above, below or approximately equal to mean, median etc...)

4.2. Computing Goals

The design and development of software to analyse the data in astronomical catalogues and produce outputs, including catalogues, is one of the major goals of this project. This project also tests and uses that software in a number of ways, such as timing grid jobs, both individually and collectively, to assess their performance and to identify occasions on which prediction and outcome do not match. The demonstration and analysis of the performance of the computing resources used in this project therefore also constitute a major computing goal of this project. As discussed in the following Subsections.

4.2.1. Demonstration of Grid Use on Data-Rich, Process Poor Problem

The types of computing problem where the computational processes are more complex and dominate the overall runtime of the program are typical of those to which grid computing in the EGI system has previously been applied. [88]

The process by which valuable information is extracted from source data in *this* project is computationally simple, as will be shown in Chapters 5 and 7, but there is a lot of data over which to iterate this process. This type of computational problem has been referred to as a “data rich, process poor” problem. As is demonstrated in Section 13.2, data access times can dominate over processing time in problems of this type.

Furthermore, the application of grid computing to a new type of computing problem therefore has the potential to demonstrate new capabilities and highlight any weaknesses in the existing grid system as are proposed by work towards the “Data Grid.” [138, 88]

As a result of the relative novelty of the pattern of use of the grid proposed in this project, novel grid job submission, monitoring and retrieval systems must be built using the existing `gLite` utilities. These systems are discussed in detail in Subsection 7.2.4.

4.2.2. Computing Metrics

As discussed in Subsection 4.2.1, this computing problem at the core of this project is of a different type to those to which grid technology has previously been applied. As a result, it is imperative that the project be carefully planned and monitored at multiple levels (e.g., calculating the processing time per catalogue entry, per file, and per grid job, which bundles many files together.) Metric and measurements are also necessary in order to understand how the software system unfolds with such a grid solution, requirement measurements (such as processing time, data access time and data volume.) These metrics are used to characterise and evaluate the performance of the system. These benchmarks can be also used to identify strengths and weaknesses in the project design and/or the grid system itself.

If any element of the project comes to dominate over the rest, or form a bottleneck which slows the running of the program, this must be identified, and steps taken in future versions to resolve these problems.

The metrics defined in this Subsection are split into three key Sections:

- Data Metrics, which are concerned with the amount of data present at various stages of the project, and how that data is organised and accessed.
- Timing Metrics, which evaluate the time taken to access, process and write the data in various software components of the project.
- Grid Metrics, which discusses issues specific to the grid, including access to grid data storage, job submission and retrieval, and limitations of the grid itself.

4.2.2.1 Data Metrics

Although Chapter 8 discusses the details of the nature and organisation of the data used in this project, a brief outline is given here for reference. Firstly, the data for this project is derived from *source data* from SDSS. That source data is first pre-processed into an intermediate state, called the *Local Catalogue*, and then processed in a pipeline to produce *Output Catalogues*. At each of these stages – source, local and output, the data is organised into many files of various sizes. The source and local files are further organised into a hierarchical file structure as discussed in Subsection 8.2.5.1.

In order to provide the user with an understanding of the characteristics of the data, these files can be assessed on the following criteria: their size, the quantity of files and the breakdown of the data structure used to store them. Where a distribution exists in these parameters, relevant quantities, such as maximum, minimum and average values for that parameter can be identified.

It is also possible that files may be missing from a dataset. These missing files may or may not affect other output data. Missing files may be due to hardware failure, for example disc failure or software failure, for example where a grid job was cancelled by the job manager, as discussed in Chapter 9. The number of missing files must be assessed, any impact on other data must be flagged, and where possible, the causes of missing data must be attributed and assessed.

4.2.2.2 Processing Metrics

This project consists of three key stages discussed in detail in Section 7.2. Those stages are downloading the source data; the Application Programming Interface (API) stage, which generates the Local Catalogue; and the Data Pipeline stage which generates the output data. Within the API and Pipeline stages, the data must be input, processed and then output again. The process of parallelisation separates the API and Pipeline stages into individual jobs.

From a perspective of acquiring processing metrics, these stages of the project can each be timed as a whole, or as individual jobs. Within each of the stages, the sub-stages can be separately timed and examined. Trends within the timing data can be plotted and observed.

From this information, the breakdown between data access and processing time can be fully characterised, and bottlenecks in the program can be identified and rectified.

4.2.2.3 Grid Metrics

Generating, submitting and retrieving thousands of individual grid jobs, accessing data through grid mechanisms and other concerns specific to the use of grid computing must be assessed. Mechanisms for monitoring grid programs are discussed in Chapters 7 and

9. The performance of the grid-specific systems can be monitored by timing in the same way as the data processing above.

Limitations on the run time of individual grid jobs demand that grid jobs must be calibrated to fit those limits. In addition, the submission, launch and retrieval of grid jobs take a finite amount of time. Therefore a clear understanding of those timings must be established before a job can be submitted.

4.3. Summary

This project has two primary elements, the astronomical and the computational. Each serves the other: the astronomical objectives guide and provide utility to the computational, and the computational element provides the tools needed to generate the data needed for astronomical studies.

From an astronomical perspective, the objectives of the project are as follows

- Develop and demonstrate a flexible system to provide optimal pointings and a score for one or more targets given the data on the stars around it and a set of observational parameters.
- Apply this system to the SDSS Fourth Quasar Catalogue to produce up to 77,429 entries in an output Quasar Catalogue of pointings for optimal differential photometry.
- Apply this system to all entries in SDSS to produce a catalogue of up to 357,175,411 entries of optimised pointings for stars, such that they might be used to observe Exoplanets.
- Analyse the output data from the Exoplanet Catalogue to identify trends in the data. This will help guide use of the data by identifying areas of the phase space of resulting data which should be emphasised by end users and those which should be avoided.

From a computing perspective, the objectives are to

- Design and implement a flexible software system capable of accepting variable input and producing appropriate astronomical results as described above based on that input.

- Enable that system to be operated in a grid environment to allow for the processing of large catalogues.
- Organise, and collate the output data, and provide metrics on its storage
- Assess the performance of that system in unit testing to identify trends and patterns that can be used to guide grid jobs.
- Assess the performance of the grid jobs in full-scale operations and compare these metrics with the predictions based on unit tests.

5. Locus Algorithm & Scoring

Differential photometry requires that the target and a number of reference stars be included together in a small FoV. These references provide an ensemble against which the target can be compared, as discussed in Subsection 2.2.3.2.

As described in Subsection 2.2.3.5, it is possible to improve the precision of the photometry achieved by using more reference stars, and by including reference stars that more closely match the target in magnitude and colour. [1] [51] Therefore, an objective of this project was to develop a system to maximise the number and quality of reference stars for a given target by adjusting the point of aim (pointing) of the telescope as set out in Subsection 4.1.1.

The Locus Algorithm, first discussed in Creaner et al., 2010, [3] is the solution to this problem that was used in this project. There are two core concepts to the Locus Algorithm. The first is that it is possible to define a locus about a point on the sky, such that a FoV centred on any point on that locus will include the original point at the edge of the FoV. Any FoV centred within the locus will therefore also include the target, and any FoV centred outside the locus will not.

The second core concept is that applying this locus to a number of candidate reference stars near to one another will cause these loci to intersect. At the points where the edge of one locus intercepts the edge of another, the set of stars which can be included in a field of view centred at that point changes.

This is illustrated in Figure 5-1. The red and blue boxes define the loci upon which a FoV may be centred to include their respective stars at the edge. A FoV centred in the pink area will include only the red star, while one in the pale blue area will include only the blue. Any FoV centred in the purple area will include both, while one centred in the black areas will include neither.

The points of intersection (PoI) between the red and blue loci, highlighted in yellow, are the points where both stars will be included at the edge of the FoV. More generally, given a larger sample of stars, it is at these points that the set of stars that can be included in the FoV changes. Therefore, the Locus Algorithm identifies these PoI.

By examining these PoI, the set of stars that can be included in a FoV centred at one such point can be compared with the set of stars included in a FoV centred at another. A scoring system is therefore required to assess which PoI would produce a better pointing. By assessing each set according to the scoring system, it is possible to determine a point at which the score reaches a maximum value.

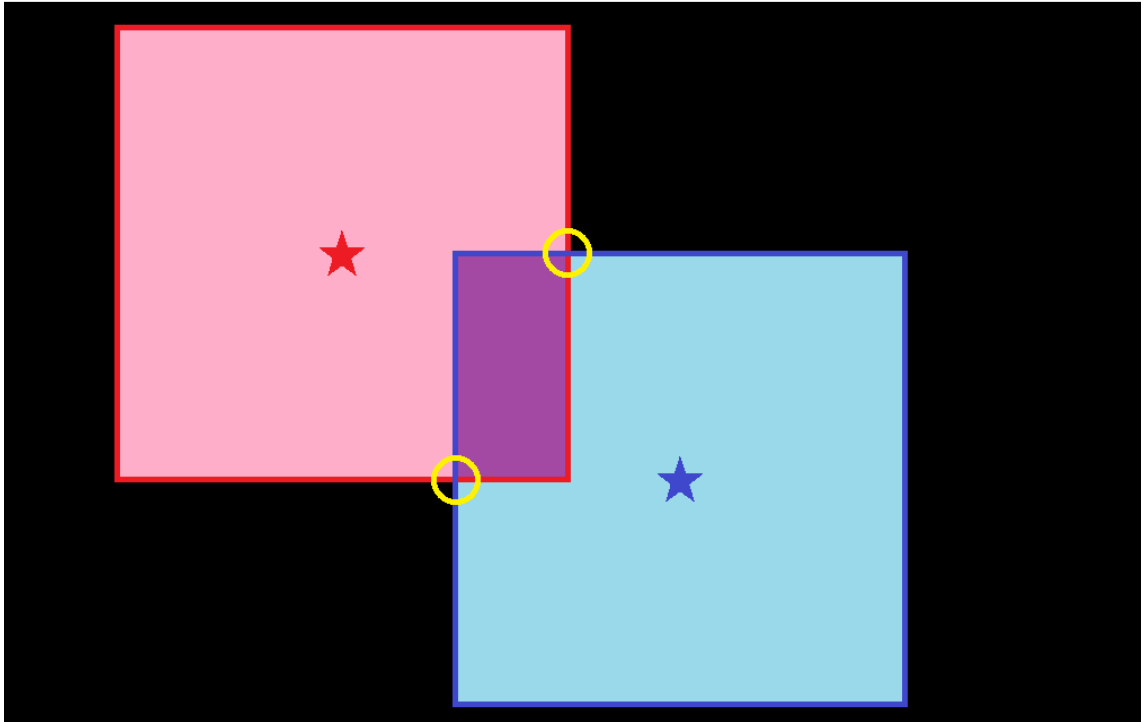


Figure 5-1: Principles of operation of the Locus Algorithm for two stars.

The Chapter first considers how to mathematically define a locus of points in the Equatorial Coordinate system. Then, a step-by-step definition of the algorithm is given with illustrations. Finally, the Chapter discusses a number of options for the scoring system, a modular component that has been developed to suit the needs of this project, and explains the choice of which system to use.

5.1. Coordinate System

Using Cartesian coordinates, and given a square FoV oriented such that its edges are aligned with the primary x and y axes, and restricting the movement of the field to x or y translations, the edges of the locus about a given point can be defined by a set of four lines, two running of constant x (parallel to the y axis) and two of constant y (parallel to the x axis), each half the size of the field of view away from that point.

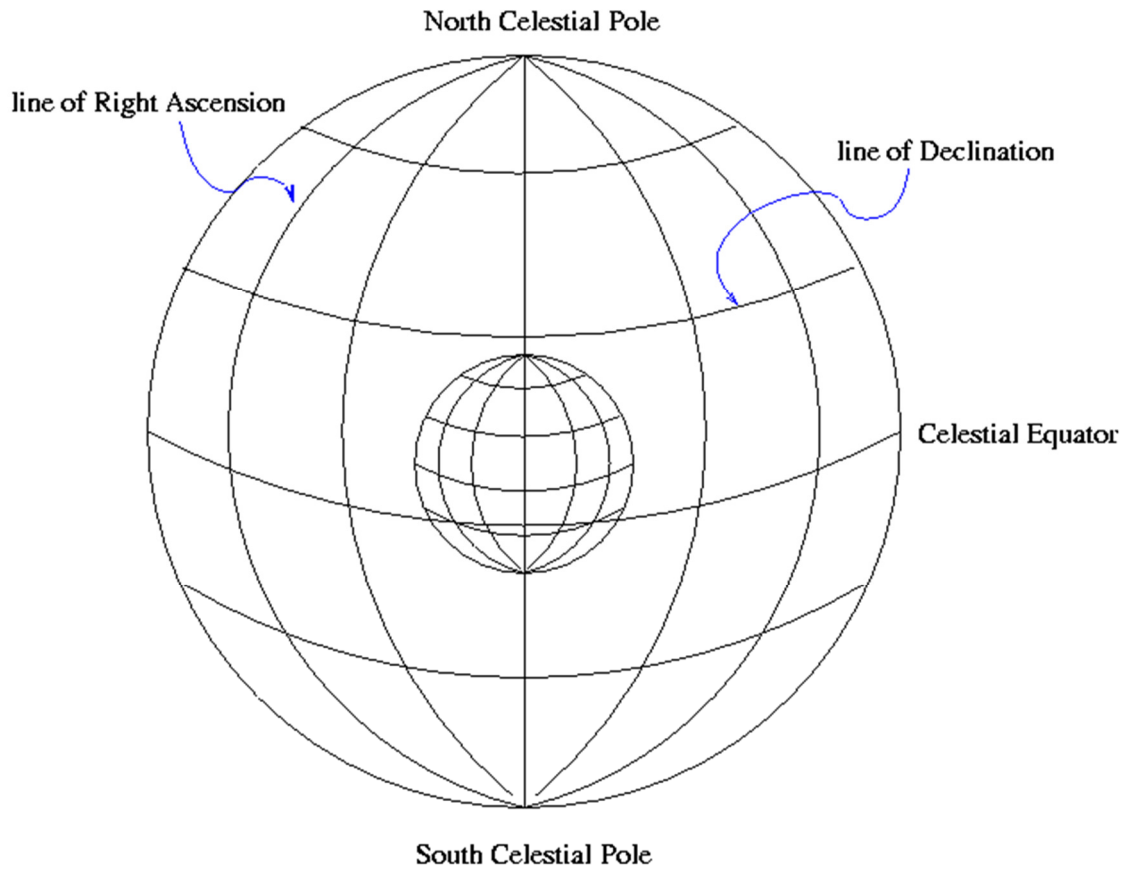


Figure 5-2: Celestial Coordinate System showing the convergence of meridians towards the pole and the consequent foreshortening of unit angles in Right Ascension. From A review of Coordinates, Redmond [139]

However, the Locus Algorithm operates in the Equatorial Coordinate system, where Right Ascension (RA) and Declination (Dec) are spherical Polar Coordinates. As shown in Figure 5-2, this means that while a unit angle in Dec remains equal to the same unit angle in true angular separation at all points on the surface, unit angles in RA are only equal to true angle (and consequently, to unit angles in Dec) at the celestial Equator. Away from the equator, lines of constant RA converge, coming to a point at the poles.

This means at Dec other than 0° , unit angles in RA are foreshortened by comparison to unit angles in Dec. The degree of this foreshortening is given by Equation 5-1.

$$angle\ in\ RA = \frac{true\ angle}{\cos(Dec)}$$

Equation 5-1: Relationship between true angle and angle in Right Ascension. [139]

By using this conversion factor, it was possible to approximate a Cartesian coordinate system with RA and Dec. This reduced the computational complexity of the Locus Algorithm by allowing a small, square Field of View (FoV) of angular side S , centred on a point (RA_1, Dec_1) , to be defined by lines of constant RA and Dec as shown in Equation 5-2.

This equation also defines the locus of points about a star located at point (RA_1, Dec_1) , upon which a FoV of size S could be centred so as to include that star.

$$RA_1 - \left(\frac{S}{2\cos(Dec_1)} \right) \leq RA \leq RA_1 + \left(\frac{S}{2\cos(Dec_1)} \right)$$

$$Dec_1 - \left(\frac{S}{2} \right) \leq Dec \leq Dec_1 + \left(\frac{S}{2} \right)$$

Equation 5-2: Definition of a Fov of size S centred on a point (RA_1, Dec_1)

For small FoV, such as the 15 arcminute FoV used in the generation of the Exoplanet Catalogue, this approximation was calculated to be accurate to within 1% for fields of view outside the polar regions (i.e. for $|Dec| < 66.5^\circ$.)

Corrections for within the polar region are more complex, and are beyond the scope of this project. These corrections are proposed as a refinement of the project in Subsection 14.3.1.2.

In addition, the definition given in Equation 5-2 ignores the fact that Right Ascension “loops around.” As a result, for example, an object at $(359.99^\circ, 0^\circ)$ would not be included in a FoV of size 0.25° with an object at $(0.01^\circ, 0^\circ)$ even though their true angular separation is 0.02° .

Since this limitation is only significant in a narrow strip of the sky around RA 0° , representing just 0.223% of SDSS detections in DR7, allowance for this was not included in the scope of this project, but is considered a refinement that could be made in Subsection 14.3.1.1.

5.2. Locus Algorithm Definition

The following is a step-by-step set of instructions describing the Locus Algorithm in detail, as it was implemented in this project. The design for the implementation of this algorithm is given in Subsection 7.2.2.1, and Chapter 10 demonstrates its application.

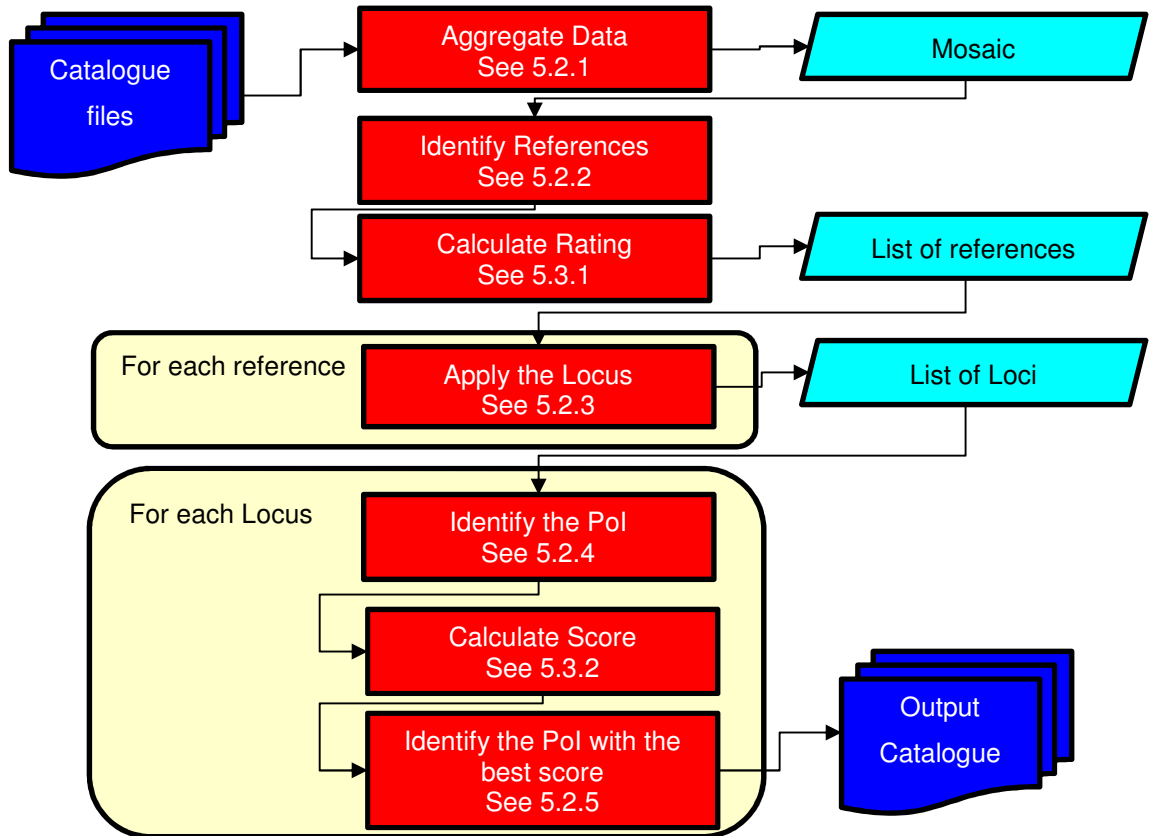


Figure 5-3: Overview of the Locus Algorithm Process. Data from the Catalogue is aggregated to form a Mosaic in memory. Candidate reference stars are identified within this Mosaic, and a Rating is calculated for each. A Locus is drawn around each where a FoV may be centred. The Points of Intersection (PoI) between these Loci are identified and a score calculated for each. The PoI with the highest score is output as the optimum pointing for that target. Each of these steps is illustrated in its respective subsection below.

5.2.1. Aggregate Data from SDSS

As shown in Figure 5-4, the data for stars around a target or set of targets will be contained in many SDSS files, and as such, many local catalogue files. The program assembles a single array from the files that cover the area of sky near to the target as

specified in the parameter file. Since this array is composed of a number of “tiles” in the form of the input catalogue files, this array is referred to as a “mosaic.”

Any star up to the full size of the field of view in any direction from a target can potentially be included in a field of view with the target. Therefore, data must be assembled from any file which includes any star within those limits.

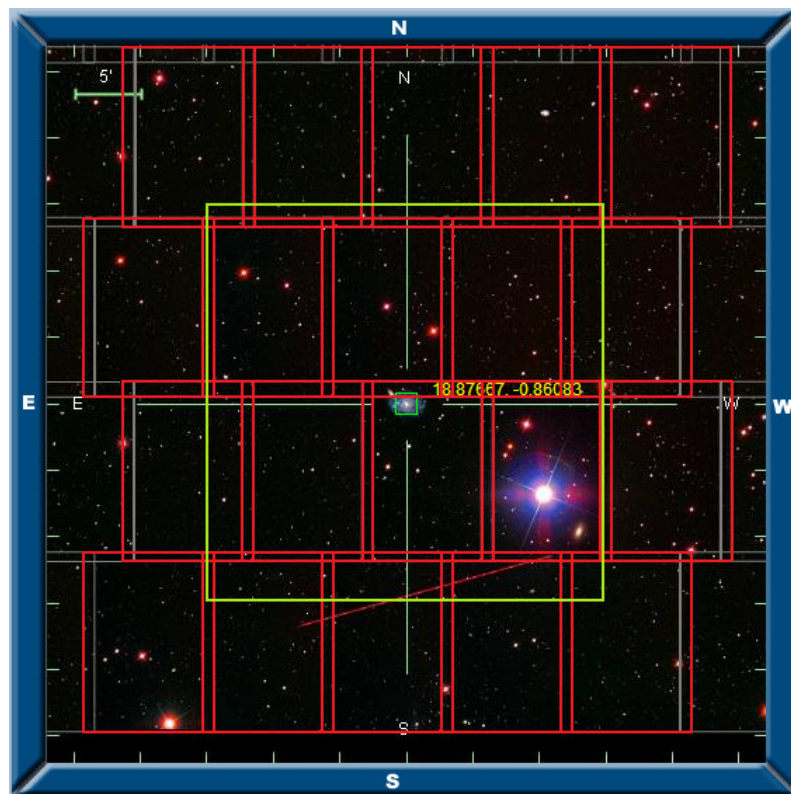


Figure 5-4: Modified image taken from SDSS Navigate image showing fields. [140] To obtain data on all stars in the green box, data from the SDSS fields marked in red must be aggregated to form a mosaic.

5.2.2. Identify Potential Reference Stars

For each target, a short list is produced of those stars which would be suitable for to use as reference stars. These stars must meet the following criteria.

- Position must be within the size of the field of view of the target.
- Magnitude must be within defined limits of the target’s magnitude
- Colour must match that of the target to within user-specified limits.
- They must be resolvable: i.e. no other star is within a user-defined resolution limit of the target.

Figure 5-5 illustrates these filters. These filters together create a list of stars that are suitable for use as references with the target, known as candidate reference stars. At this point, the first element of the scoring system assigns a rating to each candidate reference star. The reference stars are compared with the target, and a *rating* between 1 and 0 is assigned to each based on their similarity to the target using the rating system discussed in Section 5.3.1.

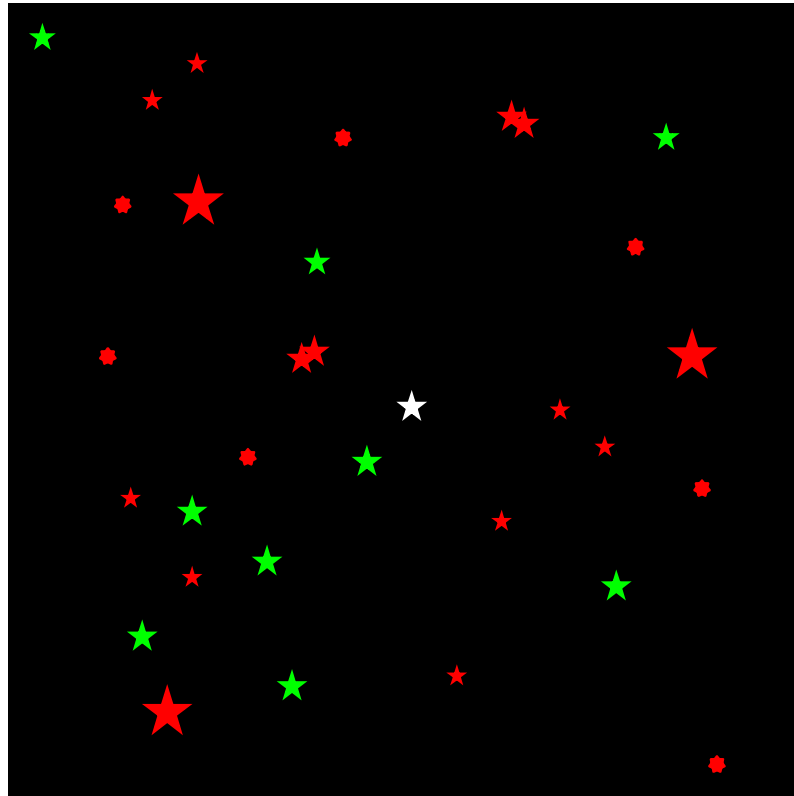


Figure 5-5: Reference stars must be identified from among the stars in the mosaic. The target is shown in white. Potential reference stars are indicated in green. Rejected stars are indicated in red. Stars are rejected if they are too bright or faint (indicated by size), if their colour indices are different to the target (indicated with 7 point stars) or if they cannot be resolved (indicated with overlapping stars.)

5.2.3. Apply the Locus to Each Candidate

For each candidate reference star, a locus of points upon which the centre of the field of view may be placed to include both the target and the reference is defined. Starting from the definition of the locus given in Equation 5-2, it follows that, for each reference, only the sides of the “box” nearest to the target need to be considered: no point on the side of the locus in the opposite direction to the target will include the target.

The locus is therefore defined as a line of constant RA and a line of constant Dec drawn from a point called the cornerpoint because it defines the corner of the Locus.

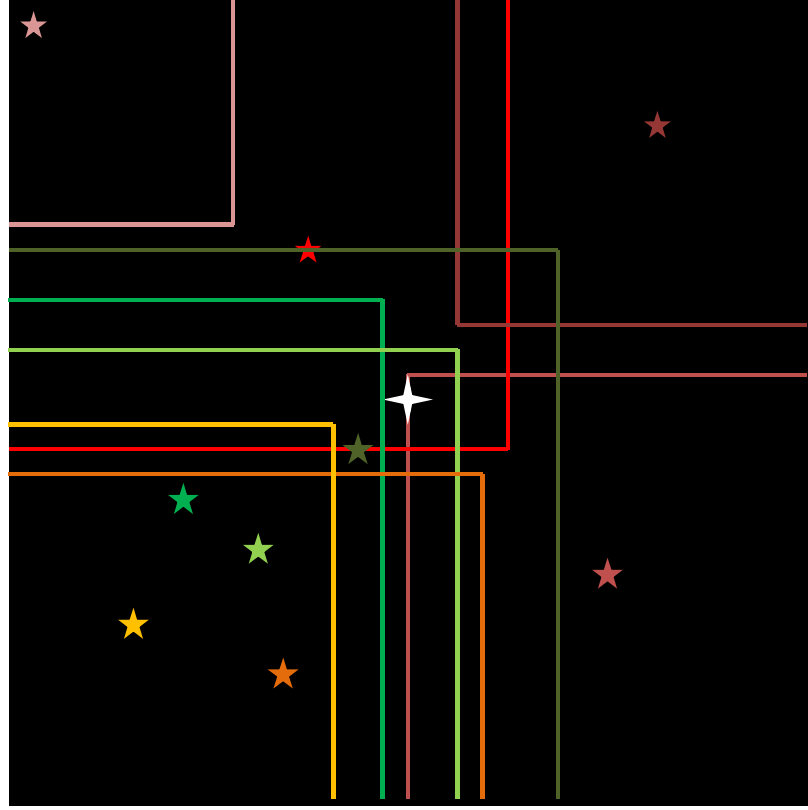


Figure 5-6: The loci are defined by assigning a pair of RA and Dec coordinates to a cornerpoint and a pair of Boolean switches which indicate whether to draw a line North or South and East or West from the cornerpoint. Each Star is assigned a colour, and the locus that corresponds to it is drawn in the same colour.

Given the coordinates of the target $(RA_{Target}, Dec_{Target})$ and the coordinates of the reference $(RA_{Reference}, Dec_{Reference})$, and a size of FoV S , the coordinates of the cornerpoint $(RA_{Cornerpoint}, Dec_{Cornerpoint})$ are defined as shown in Equation 5-3

$$\begin{aligned}
 RA_{Target} \leq RA_{Reference} &\Rightarrow RA_{Cornerpoint} = RA_{Reference} - \left(\frac{S}{2 \cos(Dec_{target})} \right) \\
 RA_{Target} > RA_{Reference} &\Rightarrow RA_{Cornerpoint} = RA_{Reference} + \left(\frac{S}{2 \cos(Dec_{target})} \right) \\
 Dec_{Target} \leq Dec_{Reference} &\Rightarrow Dec_{Cornerpoint} = Dec_{Reference} + \left(\frac{S}{2} \right) \\
 Dec_{Target} > Dec_{Reference} &\Rightarrow Dec_{Cornerpoint} = Dec_{Reference} + \left(\frac{S}{2} \right)
 \end{aligned}$$

Equation 5-3: Definition of the Cornerpoint of the Locus for a given candidate reference star

For each line, its direction is described by a binary switch – assigned a value of 0 for negative, 1 for positive. If the reference has a higher value in RA or Dec than the target, the line for the locus is drawn in the positive direction, if the reference has a lower value; the line must be drawn in the negative direction as shown in Figure 5-6.

5.2.4. Identify the Points of Intersection

The points where these lines intersect are identified. This is done by pairwise use of the RA coordinate of the cornerpoint for one reference and the Dec coordinate of the cornerpoint for another.

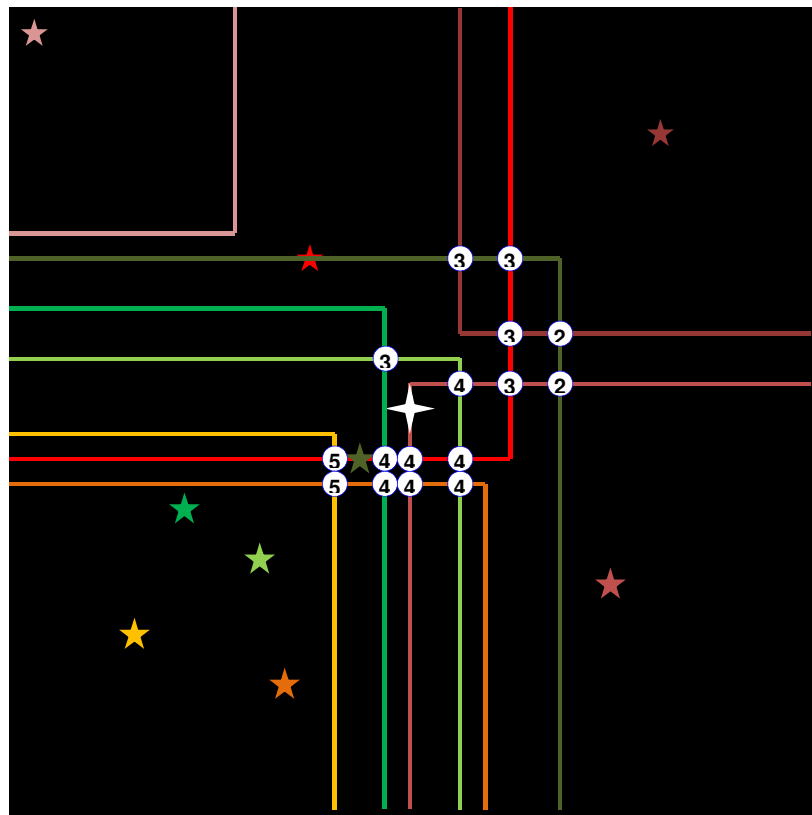


Figure 5-7: The intersection points between the lines are the points at which the score changes. For clarity, each star has been assigned a rating of 1 in this example. As a result, the score for each PoI is equal to the number of stars within a FoV of the PoI.

It is checked whether this pair of coordinates is actually an intersection point by checking the direction of those lines. For example, a line of constant Dec in the negative RA direction will not intersect a line of constant right ascension where the latter has a higher RA value than the point from which the former is drawn. The result is shown in Figure 5-7.

For each valid PoI, a score is calculated by combining the ratings of all potential references that can be placed within a field of view centred on that PoI as discussed in Subsection 5.3.2. These references are identified by determining the absolute value of the difference between their RA and Dec coordinates and those of the PoI and ensuring that value is lower than half the size of the field of view.

5.2.5. Output the Intersection with the Best Score

When the first intersection with a valid score is identified, the coordinates of that intersection and its score are put into a variable called “best intercept.” When subsequent intersections are identified, they are compared with this best intercept.

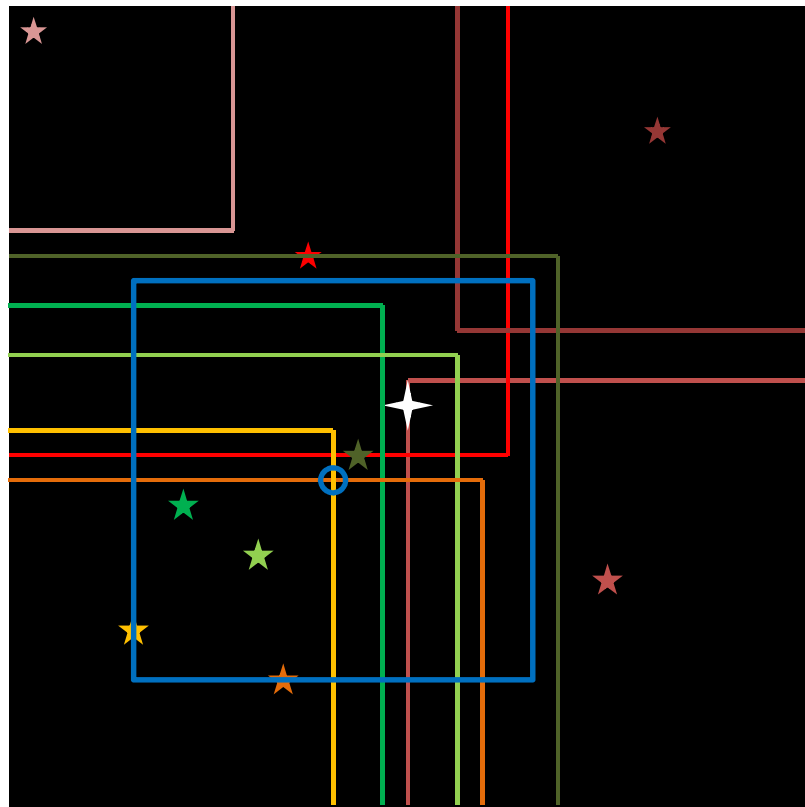


Figure 5-8: Locus Algorithm. Target: white star. Pointing & FoV: blue. Reference stars and their loci: Fully in the FoV: greens. On the edge of the FoV: yellows. Outside FoV: reds

If their score is *higher* than the best so far, the new value is put in the place of the old. When the last intercept has been identified and a score assigned to it, the best intercept is put into an output array of all the best intercepts for each target. In the event that

multiple PoI are assigned the same score, the first of those PoI to be processed will be the one that is included in the output.

5.2.6. Failed Targets

Targets for which no valid interception exists, or which have no viable reference stars are assigned a pointing with coordinates (RA: 0, Dec: 0) and a score of 0, and are considered to have “failed.” Targets may fail for the following reasons, as shown in Figure 5-9:

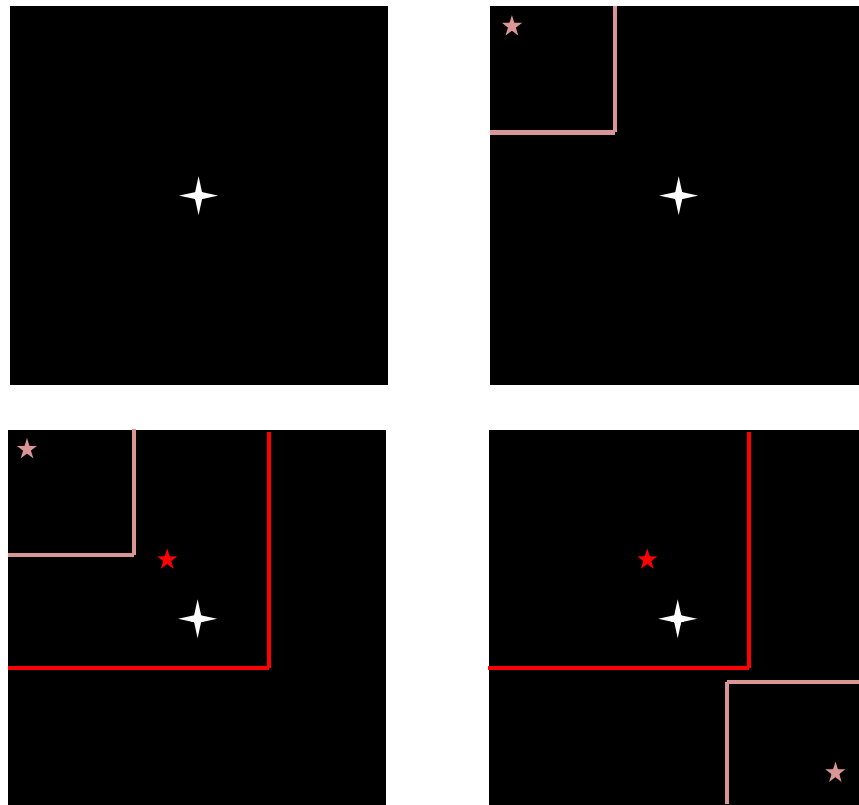


Figure 5-9: Four conditions upon which a target will fail. Clockwise, from top left: (1) a target with no references, (2) a target with one reference, (3) a target the loci of whose references do not intersect and (4) a target for which the loci of its references are nested such that they do not intersect one another.

- If there are no viable reference stars, the shortlist determined at step 5.2.2 will be empty.
- If there are no PoI detected at step 5.2.4, no score will be assigned. This will occur when:

- There is only one reference star: its lone locus cannot not intercept any other
- The reference stars are arranged in such a way that their loci do not intercept, either because they are too far apart or their loci nest within one another.

5.3. Scoring System

The scoring system used in this project has two primary functions. Those functions are firstly to identify which pointing, out of all the PoI, provides the most and best reference stars for differential photometry for a given target, and second to provide a means of comparing one target-pointing set with another.

With regards to those functions, there are two elements that make up the score for a given pointing: the *ratings*, a value between 0 and 1 assigned to each individual reference star, used to indicate how closely it matches the target as discussed in Subsection 5.3.1 and how those ratings are combined to make the overall *score* for a pointing, as described in Subsection 5.3.2.

The system used in this project was chosen from several options discussed. It was guided primarily by the domain expertise at BCO [23], and by the emphasis on broadband colour agreement as per Milone and Pel, 2011 [1] and Young, 1991 [51]. As discussed in Subsection 14.3.2, refinements to the scoring system are suitable for further study, with several proposed projects to carry out those refinements.

5.3.1. Rating System Options

In providing a rating for each individual reference star, the objective is to identify which stars most closely resemble the target. For each star, the following pieces of information must be made available about it: Its position in RA and Dec, and its magnitude in multiple filters, (u , g , r , i , z) from which colour indices could be calculated.

For the purposes of this project, the position of a reference star would be relevant only in so far as its position must permit it to be in the FoV with the target for a relatively narrow FoV such as the 10-15 arcminute fields used in this project.

For Magnitude, the user may specify a limit of magnitude to within which the reference star must be of the target. In the case of the Exoplanet catalogue, this limit was set to ± 2 magnitudes, a value chosen to permit the fainter references to achieve reliable SNR while preventing saturation of the brighter ones using the equipment at BCO [23].

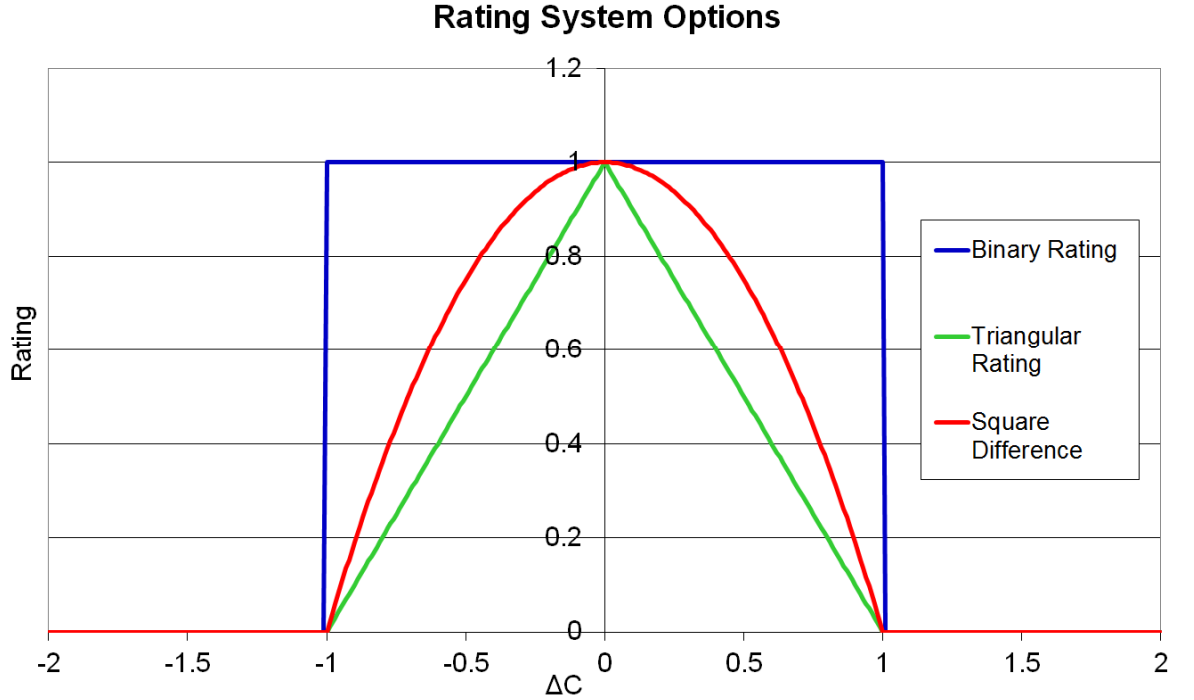


Figure 5-10: Potential rating systems. The binary scoring system, with a rating of 1 within the limit, and 0 outside. Under the triangular and square systems, rating trends from 0 at the limit to 1 at a perfect match. Triangular uses a linear progression, while square uses a parabolic progression.

Milone and Pel (2011) state that colour dependent effects cancel when stars of similar colour or “apparent spectral energy distribution” are selected as references, and that colour indices in broadband filters may be more important than spectral type in this regards. [1] Young et al. (1991) suggest that comparison stars be selected within a limit of ± 0.3 magnitude in Johnson B-V colour index. [51]

Both previous authors suggest additional factors such as small angular separation and similar or brighter magnitude as discussed in Subsection 2.2.3.5. However, it was determined, in consultation with observational experts at BCO, that for the small FoV (10-15 arcminute, much smaller than that 4-5 degree limit suggested by Young et al. [51]) and relatively small range of magnitudes (± 2 mag), that all stars which could be included in the FoV and which were within the magnitude range would be considered

equally suitable references, but that it would be necessary to differentiate between stars on the basis of the closeness of the match between their colour and that of the target. [23]

$$C_{star} = m_n - m_{n+1}$$

$$\Delta C = C_{Target} - C_{Reference}$$

$$1 - \left| \frac{\Delta C}{\Delta C_{max}} \right| = R$$

Equation 5-4: Triangular rating equation. The colour index for a given star (C_{star}) is calculated by subtracting the magnitude in the next longer filter (m_{n+1}) from the magnitude in a given filter (m_n). The Difference in colours (ΔC) is the difference between this value for the target (C_{target}) and that for the reference ($C_{reference}$). The rating (R) is then given as a value between 0 and 1 by subtracting a normalised, absolute value given by dividing ΔC by the maximum permitted ΔC from 1.

$$1 - \left(\frac{\Delta C}{\Delta C_{max}} \right)^2 = R$$

Equation 5-5: Square Difference rating Equation. Difference in colours (ΔC) is calculated in the same way as in Equation 5-4, but instead of subtracting a normalised value of ΔC , a normalised, squared value is used.

As a result of these guidelines, the user may also specify a limit to within which the colour of the reference star must match the colour of the target. For the Exoplanet catalogue, this was set to +/- 0.1 magnitudes. These limits were selected to provide a baseline well within the limit suggested by Young. [51] Within that limit, there were several options as to how to assign ratings to reference stars, as shown in Figure 5-10.

The Binary scoring system only distinguishes between references that are within the limit and those outside it: no further distinction is made within that limit. The square difference and triangular scoring systems are designed to make this distinction by assigning a rating of 0 to a reference at the limit, a rating of 1 to a reference that is a perfect match for the target on that colour. Between those limits, the triangular system, based on Equation 5-4, uses a linear progression from 0 to 1, while the square difference system, based on Equation 5-5, follows a parabolic progression.

Of the three options available, the triangular rating system was selected. It was chosen over the binary system because it reflects the design goal whereby a reference which is a perfect match for the target is given a better rating than one which “barely passes.”

$$\begin{aligned}
 C_{star\ l} &= m_n - m_{n+1} & C_{star\ s} &= m_{n-1} - m_n \\
 \Delta C_l &= C_{Target\ l} - C_{Reference\ l} & \Delta C_s &= C_{Target\ s} - C_{Reference\ s} \\
 1 - \left| \frac{\Delta C_l}{\Delta C_{max}} \right| &= R_l & 1 - \left| \frac{\Delta C_s}{\Delta C_{max}} \right| &= R_s \\
 R &= R_l \times R_s
 \end{aligned}$$

Equation 5-6: Calculation of Rating. For each star (Target and Reference) three magnitudes of increasing wavelength (m_{n-1} , m_n & m_{n+1}) are used. This permits the calculation of two colour indices: C_s calculated from the shorter wavelength pair (m_{n-1} & m_n) and C_l calculated from the longer wavelength pair (m_n & m_{n+1}). From these, two corresponding ratings (R_s & R_l) are calculated for each star as per Equation 5-4. R_s & R_l are multiplied to calculate the final rating R .

These methods apply where a single colour index is used. However, this project makes use of two colour indices: to optimise photometry for a given SDSS band, both the colour indices generated by comparing that band with its neighbours (e.g. g-r and r-i for the r band) are used, where both are available (u and z each only have one neighbour.)

Illustration of the range of values of rating for a reference based on two colour ratings combined by multiplication

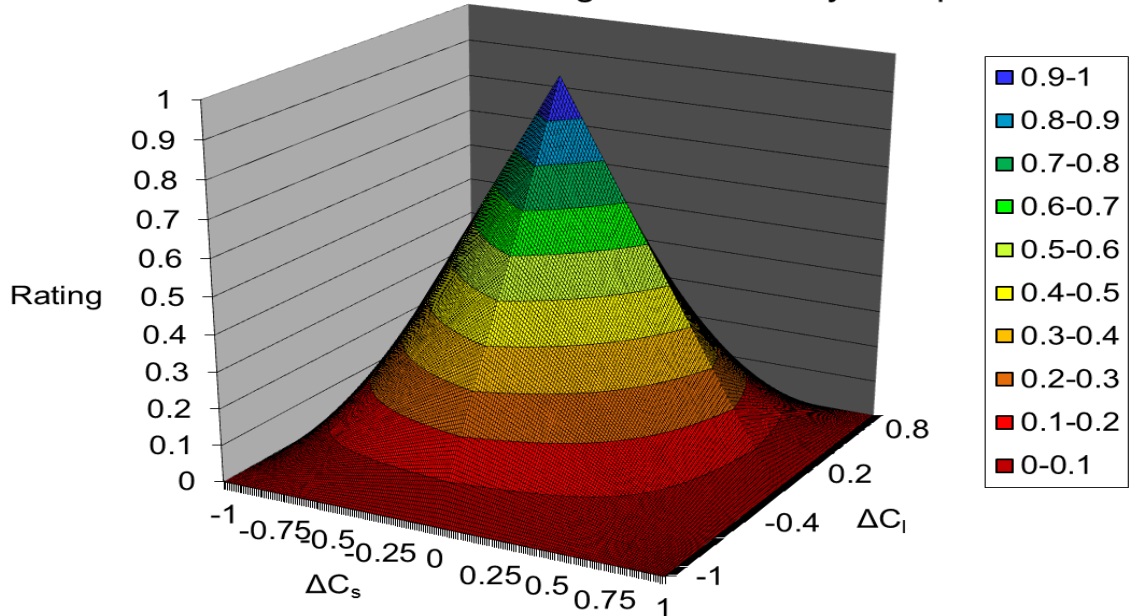


Figure 5-11: The Witch's Hat graph. This graph demonstrates the distribution of rating against difference in two colour indices calculated as shown in Equation 5-6

In order to combine the ratings from two different colour indices, while continuing to restrict the range of rating between 0 and 1, the ratings were combined by multiplying them together, as shown in Equation 5-6.

This method of combining the ratings creates a distribution of rating against the two colour indices which shows a sharp peak for references which exactly match the target, as shown in Figure 5-11, a pattern described within this project to as the “Witch’s Hat graph.”

5.3.2. Combining Ratings into Scores

Once the ratings for each potential reference star are calculated, the algorithm steps through each potential pointing and combines the ratings from the reference stars that are included in a field of view centred at that pointing.

For this project, the ratings for each star that could be included in the field were added together as shown in Subsection 5.2.5. This provides a mathematically straightforward solution to the problem of combining ratings into scores.

Meta-Analysis of the output as discussed in Chapter 12 show that scores in the Exoplanet catalogue are highly variable, with a mean of 6.74, but a distribution that showed a long tail, with one target showing a score of 117.

In ensemble photometry, each reference may be used as a target in its own right, and inter-compared with the other. By being similar to the target, the reference stars are also similar to one another. In the search for Extrasolar Planets, all stars can be considered potential exoplanet hosts. Taken together, these facts mean that additional reference stars continue to add value to a pointing even as the number of references becomes very high.

However, further refinement to the method of combining ratings into scores may be profitable, for example, to make allowance for a “Law of diminishing returns” as discussed in Subsection 14.3.2.3.

5.4. Conclusion

The Locus Algorithm consists of a series of steps to identify an optimum pointing for a given target, and to assign a score to that pointing. As discussed above, the following assumptions are made:

- FoV used are small
- FoV are square
- FoV are aligned such that their edges align on the North/South and East/West axes
- FoV may only be translated North/South or East/West

As shown in Figure 5-3, the steps required to complete the Locus Algorithm for a given target are

- Aggregate the data needed for all potential reference stars from a source catalogue, tiling multiple files together to create a mosaic if necessary
- Create a short list of candidate reference stars
- Assign ratings to each candidate reference star using the “Witch’s Hat” rating system as shown in Figure 5-11 and Equation 5-6.
- Apply the Locus to each candidate, allowing for corrections for spherical coordinates
- Identify the Points of Interception (PoI) between those Loci.
- Select the optimum pointing by calculating a score for each PoI, and designating the pointing with the highest score the optimum pointing.

To accomplish all of the goals of this project, the Locus Algorithm must be iterated many millions of times. Chapter 7 discusses the design of the software and hardware solution required to implement this algorithm, on one target or many. Chapter 9 discusses the practical implementation of this project based upon that design. Finally, Chapter 10 shows a fully worked example of this system in operation on a single target star, SDSS J113824.40+483457.8.

6. Project Design Concepts and Approach

This Chapter describes the core concept and principles of design used throughout this project, and the approach used to put these concepts into practice.

Design Concepts are a set of principles of design, which guide a designer in making design choices, regarding the order in which software components should be designed and developed, how these components should be separated, and how they should be connected to one another.

Design Concepts are considered under two headings in Section 6.1:

- Design Strategies, referring to the pattern in which component parts of the project were identified, in what order they should be developed, and at what level of design input and output parameters should be determined.
- Design Philosophy, referring to a set of principles upon which design decisions were made, such as whether to focus on maximally-optimised software for a single purpose, or whether to allow for extension to other purposes.

The Design Approach refers to the method, including an explanation of choice of the preferred design strategy used in this project to produce the final design of the project as shown in Chapters 7 & 8. This method is further divided into two components in Section 6.2

- Design Process: the series of steps used to manage the design and development of the project
- Design Techniques, a set of tools used in this process to provide consistency in the design process

These concepts and approaches were used to define all aspects of the project, including software and data, as well as the approach to the scientific solution. The resulting design of the project is described in Chapters 7 & 8.

6.1. Design Concepts

Two major design strategies are explained in Subsection 6.1.1: Top-Down design, whereby high-level elements of the design are recursively broken into smaller elements

until the implementation of that element is clear, [141] and Bottom-Up design, wherein the most basic components of the design are defined first, and assembled to make the system. [142]

The Design Philosophy Subsection (6.1.2) refers to two concepts which underpin the design of this project: Flexibility (e.g. allowing variable input parameters) and Extensibility (e.g. allowing the software system to be used with other source catalogues.) Together, these requirements demand the application of two other principles: Modularisation of software to allow new or replacement modules to be added to the software solution (e.g. the scoring system) and a Layered data structure to permit abstraction of the data from the source. (e.g. the Local Catalogue.)

6.1.1. Design Strategy

There are two basic strategies applied in software design, Top-Down and Bottom-Up. [143] It is often not practical to entirely depend on one or the other of these strategies. [144] Instead, these strategies are often used together, as was the case in this project. [142] Each of these strategies is described and the significance of that strategy to this project discussed below.

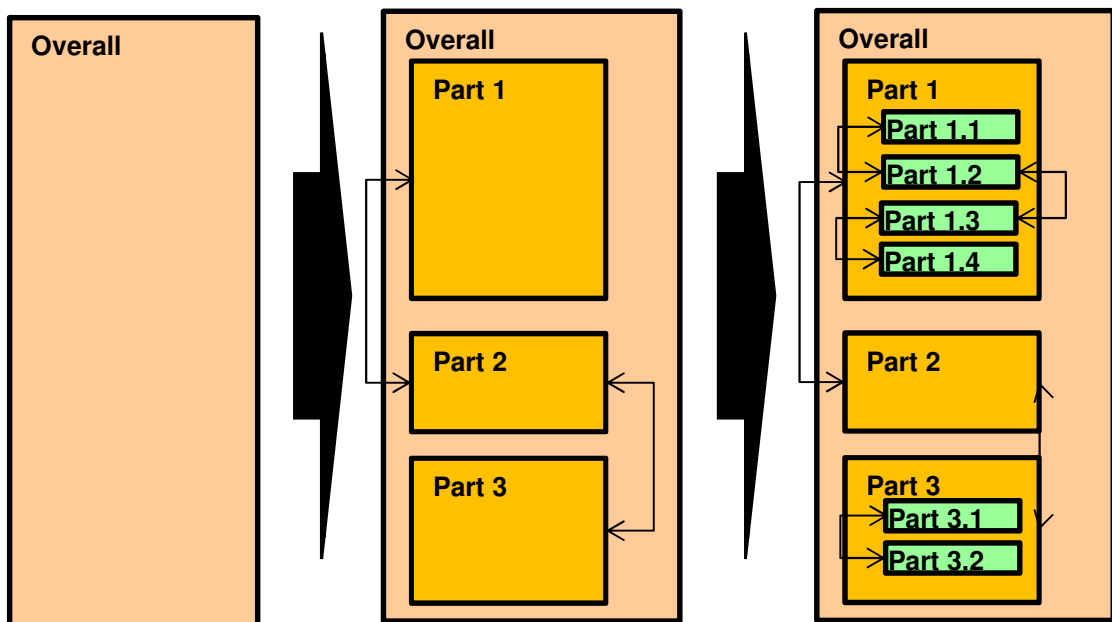


Figure 6-1: Illustration of Top-Down design concept. The overall task is broken into components recursively until it can be coded. Note that not all components require the same number of steps to refine as shown in the case of Part 2 in the illustration above.

The Top-Down design strategy requires a description of the complete system as a starting point, which may then be *decomposed* into subsystems. This process is repeated until a desired level of detail is achieved. This level of detail is set such that implementation of the lowest level component is easily understood. [141] The recursive nature of this strategy is referred to as “stepwise refinement.” [144] This process is illustrated in Figure 6-1. The advantages of the Top-Down approach are that it allows for more control over inputs and outputs to components of the project. [142] The drawbacks are that it does not readily lend itself to early testing, as some necessary components may not be available. [142]

The Bottom-Up strategy, by contrast, does not require a complete description of the system before implementation can take place. [142] Instead, basic components are created which can be assembled to allow higher levels of abstraction (cf. Subsection 3.3.1.1.) [144] These components can be assembled in any order to create the complete system. The strength of the Bottom-Up approach is that it is easier to test early and it encourages the development of reusable components. [142] A major disadvantage is that it can require an intuition on the part of the designer to identify which components will be needed. [144]

In this project, the Top-Down strategy was emphasised, as the overarching design of the project was defined from an early phase as detailed in Section 7.2. This allowed it to be decomposed into subtasks, each of which could be developed separately. In addition, much of the software for this project was to be written from scratch, which suggests a Top-Down approach as per Jalote, 2005. [144]

However, the design of this project incorporates some Bottom-Up elements. The use of existing libraries and data structures defines some low-level components. For example, the project used the SDSS catalogue, which is stored in FITS files. This necessitated the use of the FITSIO library, as discussed in Subsection 3.3.2.2. The requirement of these low-level components can be considered elements of Bottom-Up design. [144] Jalote (2005) also suggests that later iterations in an incremental development process, as discussed in Subsection 6.2.1 can be considered characteristic of a partially Bottom-Up type approach, as existing components must be used to complete the new design. [144]

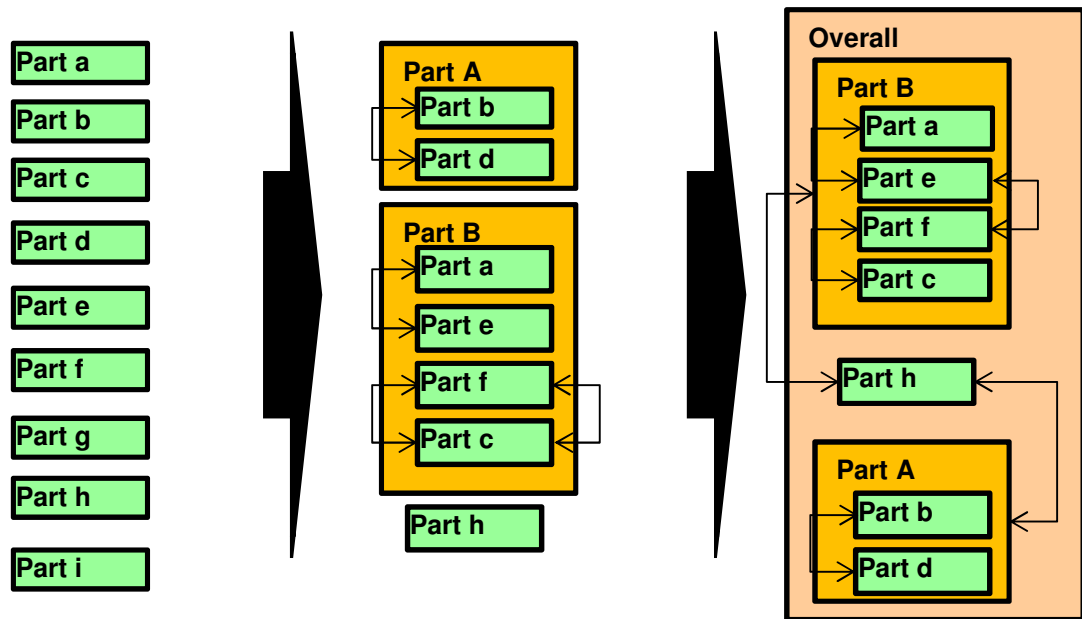


Figure 6-2: Illustration of Bottom-Up design concept. A number of components are created which are assembled into more abstract components and eventually used to form the overall program. This diagram also highlights one potential issue with Bottom-Up design. Components (e.g. parts g & i in this diagram) may be designed and developed but are unused in the final design. [144]

This Thesis generally describes the design of the project from the Top-Down perspective. Bottom-Up design considerations are usually treated as constraints to this design and are described as such where appropriate. The overall design of the project, as explained in detail in Section 7.2, reflects this approach.

6.1.2. Design Philosophy

As discussed in Section 4.3, a key design goal of this project was to maximise the flexibility of the software developed in the course of the project. Flexible design in this case would allow for the use of the software under various operational or observational conditions. For example, the FoV size discussed in Subsection 5.2.1 is variable from observatory to observatory. This required that the program be designed to allow as many of the criteria used in the analysis to be input by the user as parameters. These parameters could be further subdivided into command line arguments (for short parameters) or parameter files (for large sets of parametric data.) The parameterisation system is described in Subsection 7.2.3.

Additional flexibility beyond the scope of parameterisation is possible through extensible design. Extensible design exists where a system can have new capability added without little or no effect on the operation of the rest of the system. [145] Extensibility depends strongly on a clear overarching design, which allows for small, incremental additions to be made to the program by changing individual components. [146] A number of possible extensions to this project are considered in Subsection 14.3. Allowance for these possibilities is built in to the design, primarily through Modular Design and Data Layering, discussed below. [145]

Modular design means that software and data components can be modified or replaced individually without necessarily requiring that other components be extensively modified. [147] Modular components additionally lend themselves to being reused at different stages of the project when a similar function is needed for a different purpose. [144] The components of a modular design require a clear overall framework in which each is connected with other modules. These connections should be as simple as is practical, to minimise the coupling between modules. [148]

As an example, the scoring mechanism, for which there were several options, as discussed in Section 5.3, is implemented with a modular function within the program called `scoring_mechanism` as discussed in Subsection 7.2.2.1.1. The input to this function is a structure containing the magnitudes for the star and target, and the output is the rating for that star. So long as these inputs and outputs are maintained, this module can be replaced with another which implements a different scoring system.

Related to the framework joining modules together is data structure. Hamlet and Maybee, 2001 explain that data structures may be left vague at early stages of a Top-Down design, only being detailed when a module requires them, or may be pinned down at the start. [147] As this project required the use of FITS files from SDSS, much of the structure of source data, including the file storage structure defined in Subsection 8.2.5.1, was known from the outset. This Bottom-Up element impacted the design of the data structure used in the project. Some elements of the data structure used would be inherently dependent upon the data structure of SDSS. To provide for extensibility to other catalogues, it was necessary to create layers of data abstraction between source data and the data used in the data processing pipeline.

The primary separation of data layers in the project was created by means of an Application Programming Interface (API), as defined in Subsection 7.2.1. The source data from SDSS was extracted into a local format that can be used by the pipeline without being dependent upon the data structure of SDSS. Additionally, it is possible, as proposed in Subsection 14.1.5, to extract data from other catalogues to this local format such that the pipeline can be used with minimal changes. Both of these data formats are discussed in Subsection 8.3.1.

Since both the internal and directory structure of the SDSS Files was well defined at the outset of the project, it was decided to develop a rigorous layout for much of the remaining directory structures (shown in Section 8.2) and the various file types to be used (shown in Section 8.3) at an early phase of the project.

6.2. Design Approach

The design process used in this project emphasised a Top-Down approach as discussed in Subsection 6.1.1. Within that strategy, the design process primarily followed a breadth-first approach as illustrated in Figure 6-3, where all components at a given level of abstraction are designed before any components of the next level down. [149]

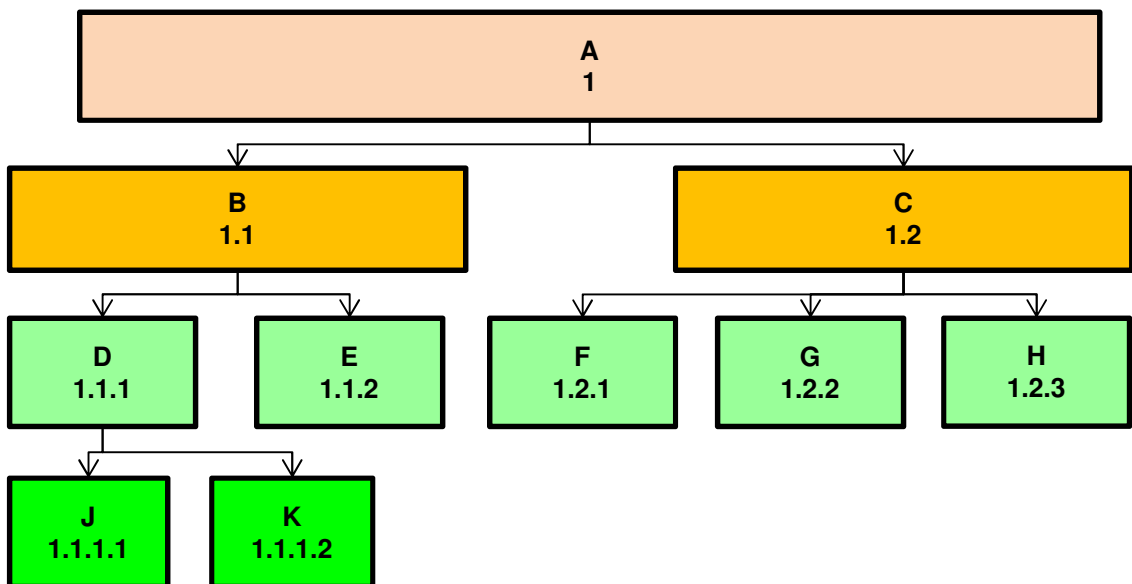


Figure 6-3: Top-Down design in breadth-first and depth-first modes. Breadth-first solutions follow the path indicated by the letters in the alphabetical order (I excluded for clarity.) Depth-first strategies follow the decimal pattern (e.g. 1.1 comes ahead of 1.1.1, which comes ahead of 1.2.) [149]

Within this framework, each component of the project was designed and developed by means of a cyclical and evolutionary design process which is described in detail in Subsection 6.2.1. Cyclical design refers to the notion that design and development are interwoven processes, whereby the project is first designed, then developed, then tested, and the results of that test feed back into the design. [150] Evolutionary design refers to the concept whereby software can be modified in small increments to add additional features without impairing existing functionality. [148] An example of a component of this project which was developed using the evolutionary process is the Parameterisation software, shown in Subsection 7.2.3. Each of the three elements of that software was developed as an incremental variation on the previous ones.

These two elements of the design and development process required a robust set of design techniques which were used to record the software design as it evolved in response to test results and evolving design requirements as described in Subsection 6.2.2. The most critical of these tools were a software design document template used throughout the development process and a version control mechanism which allowed for rollback when needed.

6.2.1. Design Process

The design and development cycle used in this project was based upon the Iterative and Incremental development method [150], adapted to the particular needs of this project. Crucial to the work cycle of this project was the repetition of this cycle over multiple iterations and at multiple levels. The development steps used in this project were Design, Develop, Test and Analysis as shown in Figure 6-4. [144]

- Design refers to the process of creating a detailed description of the design object (in this case, usually software or data structures) [151]
- Development, in the context of this cycle, incorporates two major elements: the application of the design and development process to the next level down in the design, and the implementation and coding of the design at the lowest level. [150]

- Testing refers to the process by which the design object would be used in a simulation of its designed purpose, the results of which could be analysed to determine if it was working correctly. [144]
- Analysis of the test results allows the developer to determine whether the design object meets its requirements. [144] The various levels and forms of testing used in this project are discussed in detail in Subsection 9.1.2.

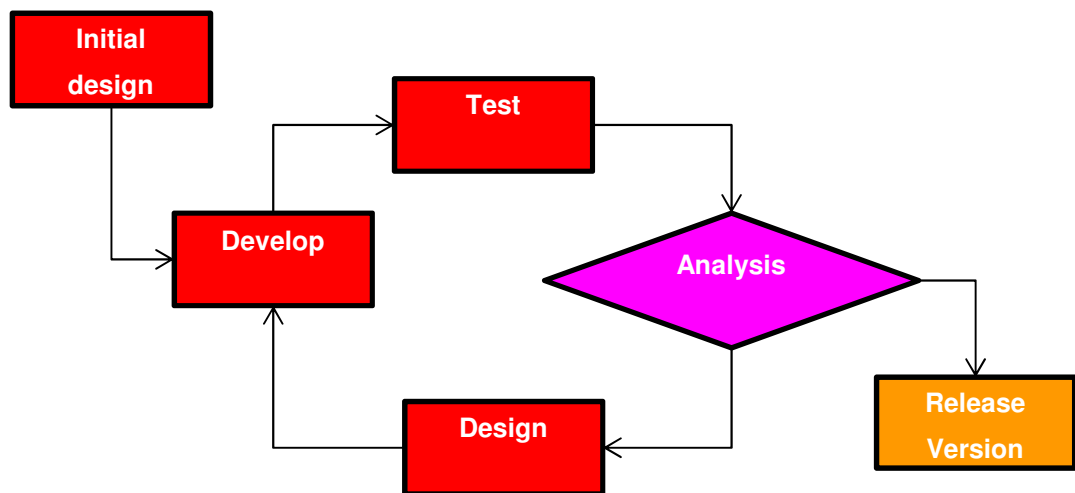


Figure 6-4: Design and Development Cycle

Starting from the highest levels, the initial designs took the form of formal software design documents as shown in Subsection 6.2.2.1. These documents were created in a Top-Down method where each subsequent level showed the detail of operation of the higher level.

On a practical level, the design documents were used as an internal reporting tool within the SCG. The formal design documents would be iterated down to functional level, below which an informal design would be sketched by the developer. The developer would then develop and code this design and test it in a functional test as described in Subsection 9.1.2. Typically, there would be refinements to this design and this process would cycle between design, development and testing on a very short timescale.

At the end of this process, when the functional level design was ready for release, the accumulated design changes would be incorporated into the higher level design document. The developer would then repeat the process at the next step in the higher level design.

6.2.2. Design Techniques

The design of this project mandated the development of a set of consistent design techniques to be used within the SCG for recording the design at each stage of development and for clarity in reporting.

An internal design document style was developed to meet the specific goals of this project. This design style included defining specific terms as they are used in this project, and a set of colour coded design symbols derived from international standard symbols [102] to ensure internal consistency in terminology. This set of symbols is shown in Appendix B.

The cyclical design and development process together with the evolutionary design of the project mandated rigorous version control. This was implemented on all design documents and code as described in Subsection 6.2.2.2.

6.2.2.1 Design Documents

During the development of the project, design documents were developed using the design terms and philosophy outlined above. Each design document consisted of two to four pages describing in brief the component of the project that it was intended for. In keeping with the Top-Down design strategy, several layers of design documents would exist for the components of the project, as each subsequent layer explored a deeper layer of the project in finer detail.

In practice, the layers of the design were not always consistent: in keeping with the evolutionary design approach, it was possible that the design of a module would become too complex to be clearly expressed in a single document. This would trigger the creation of a new layer of the design, with the lower levels being abstracted to individual documents, while the higher-level design was simplified.

Each design document followed a template similar to the one shown in Figure 6-5. Each document began with a header which included the document type, the name of the document, its version number and a date on which it was written. It then listed any inputs to and outputs from the program, function or file that the document referred to.

A brief, one paragraph description of the project component would then be given. Finally, a step-by-step description of the program or file was listed in bullet point form. The level of the design module changed what these bullet points would describe: e.g. for the highest order documents, the instructions were phrased in terms of the programs executed, while for lowest level documents the bullet points were in pseudo-code.

For data design documents, the data components were listed in the order they appeared in the file or array. The data type and names of each piece of the data are listed in courier format. Where arrays of the same type of data exist, they were listed in an indented format, usually with a number before them stating the number of elements in that array.

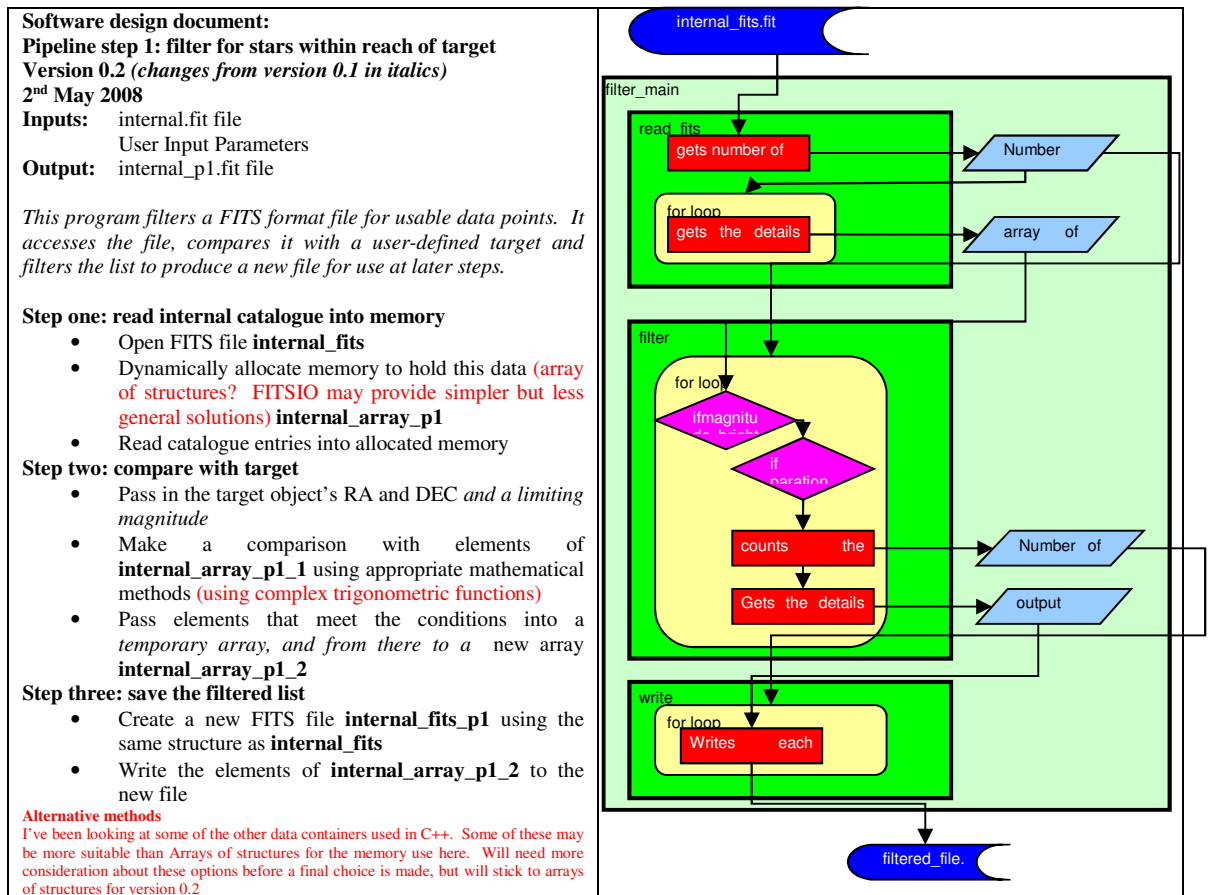


Figure 6-5: Software Design Document Example: an early version of the filtering process by which candidate reference stars are identified. The current version of this design forms the basis of Subsection 7.2.2.1.1. Note the informal notes in red text which indicate design decisions which were outstanding at this point in the development process.

The final element of a design document was the flowchart. The design document used the symbols described above to give a visual representation of the flow of data or the progress of the program.

The design of this project, based on these software design documents, is shown in Section 7.2. The document structure of Sections and Subsections are used there to show the hierarchy of the design of the project in the Top-Down, depth-first manner illustrated by Figure 6-3.

6.2.2.2 Version Control

The interleaving nature of design and development described in Subsection 6.2.1 made version control essential, both for the designs that guide the development and the software developed. A simple but robust implementation of version control was used in this project.

All files would be headed with an appropriate comment (marked in code where appropriate) giving their version number (as shown in the software design document, shown in Figure 6-5.) Each file would also incorporate its version number into the file name (for example `filter_0_2.doc` is the file shown in Figure 6-5.)

At each stage of the project's development, refinements were made and the design evolved with the needs of the project. In order to allow changes to be tracked easily, the project's design documents were themselves designed with a simple method for showing where a change had been made. Changed components were shown with *italic text* in software design documents. Changes in code were marked with appropriate comments.

The first number in the version number was only changed for major overhauls of the file, while version subheadings were changed regularly as minor changes were made.

6.3. Summary

The design of this project emphasised a Top-Down strategy. This means that an overarching design was completed first, with components developed later. These components were themselves designed before their subcomponents in a process of stepwise refinement. Bottom-Up elements entered the design strategy in the form of libraries which were needed and language or environmental limitations which had to be met. These Bottom-Up elements are treated as constraints to the design.

It is intended that the design of this project be as flexible as possible while maintaining functionality. As a result the following elements of design philosophy were adhered to.

- Flexibility at a user level was implemented by parameterisation where possible, in the form of command line arguments or parameter list files which the user could input.
- Extensibility was built into the code such that programmer-level modifications to specific elements of the project were possible without a radical redesign being needed.
- Modular design combined with a clear data-flow framework to which modules could be attached permitted components to be swapped out to allow additional functionality or reused in other stages of the project.
- Data Layering provided a layer of abstraction between source data and internal data that allowed for changes in source data without changes to internal software.

Where possible, the design approach used in this project followed a breadth-first model, where a full description of all elements of a layer of abstraction would be developed before the next layer of the Top-Down structure could be designed. At each level, a component would be designed using the cycle of design → develop → test → analyse → design. Feedback from the analysis could be combined with evolutionary changes in requirements to guide the next version of the design.

These designs were recorded in a suite of formal design documents. These documents were structured to adhere to a template as illustrated by Figure 6-5. Both design documents and code were subject to rigorous version control to ensure changes could be clearly tracked and to permit rollback where needed.

The design of software shown in Chapter 7 and the data structures shown in Chapter 8 is derived from these documents, and follow the principles of design outlined in this Chapter.

7. Project Design

This Chapter presents the final software and data flow design of the project as it was implemented in the course of this project. The design shown here is the end result of the evolutionary and iterative design and development process described in Subsection 6.2.1. The discussion of this design is split into two main Sections: the Design Requirements & Constraints and the Overall Software Design.

Section 7.1 is a summary of the goals, constraints and requirements of the project as discussed in Chapters 2-6. As discussed in Subsection 6.1.1, while these constraints can be considered to impose an element of Bottom-Up flavour upon the design of this project, the overall design of the project emphasised a Top-Down software design strategy in a breadth-first mode as discussed in Subsection 6.2.1. [144]

The resulting design of the project is laid out in detail in Section 7.2 in a Top-Down, depth-first pattern as defined in Figure 6-3. The depth first approach eases comprehension of the design, as each high level Subsection (e.g. 7.2.1) refers to a module and the lower level Subsections (e.g. 7.2.1.1 and 7.2.1.2) refer to components of that module.

The data management design used in this project, and which provides further context for the overall design presented here, is explained in detail in Chapter 8.

7.1. Design Requirements & Constraints

The design constraints of this project constitute an element of Bottom-Up design in this project as discussed in Subsection 6.1.1. [144] The details of the design requirements are discussed in length in Chapters 2-6, and are summarised here to provide context for the design of the project as implemented later in this Chapter. This perspective is discussed here under two subheadings: Astronomical Specification and Computational Limitations, and summarised under Consequent Software Design.

7.1.1. Astronomical Specifications

The Astronomical goal of this project, as detailed in Section 4.1, is to create a catalogue of stars suitable for use in the search for extrasolar planets by the transit method using

differential photometry, together with the ideal pointing for each star to maximise its utility for this technique.

The requirements to achieve this goal can be summarised under the following four headings:

- A source of suitable astronomical data including star positions and magnitudes
- A method to generate pointings for the observation of those stars for a given FoV
- A mechanism by which these pointings can be compared with one another
- A system for recording and accessing those pointings

The source data used for this project is the SDSS Catalogue as discussed in Section 2.3. The SDSS Catalogue is available through the CAS and DAS systems as discussed in Subsection 3.3.2.1 [111]. The DAS access method was chosen, which provides the SDSS data in the form of a collection of FITS data table files [111]. FITS file access requires the use of the FITSIO library. [129] The software solution must therefore be developed in a language which has a robust version of this library available to it as discussed in Subsection 3.3.2.2.

The method developed to generate pointings is the “Locus Algorithm,” discussed in detail in Chapter 5. Each iteration of this algorithm requires access to the data for the region of sky surrounding the target star as shown in Subsection 5.2.1. The algorithm was iterated many times to create the Exoplanet catalogue as discussed in Subsection 11.4, using the entire SDSS catalogue as an input. Accessing files or network resources can be a bottleneck in this process, so a design goal was set to minimise the number of these operations.

In order to identify the optimum pointing for a given target, and to allow the pointings for different targets to be compared with one another, a scoring system was developed as shown in Section 5.3. As there are several options for scoring system, this component was designed as an interchangeable module within the software solution.

It was decided to use FITS for storage of the output. FITS has long been the standard file type among the astronomical community and is internationally used to store both

data tables and images. [129] Furthermore, the use of FITS for both input and output of data meant that this did not place any additional design constraints (such as a requirement for additional I/O libraries) on the project.

Although FITS is the *de facto* standard for astronomical data transfer, its origins in the late 1970s [152] impose limitations on its adaptability to more advanced challenges, such as the limited character set (US-ASCII) and inadequate support for large, distributed datasets in a robust, and transparent manner. [153]

The Hierarchical Data Format, Version 5 (HDF5) was adopted by the Low-Frequency Array for radio astronomy (LOFAR) in 2011, [153] and authors such as Price et al (2015) [154] and Thomas et al (2015) [153] suggest it would be a suitable replacement for FITS in the medium-to-long term. The advantages of HDF5 include much faster I/O and better compression of the data. [155]

In the short term, however, Price et al (2015) [155] highlight a lack of widespread support for HDF5 in the field, with particular emphasis on the lack of data reduction and image viewing packages. This lack of adoption in existing applications mandates that this project must continue to use the existing standard (i.e. FITS) to provide for wider accessibility, though future implementations such as those discussed in Chapter 14 may make use of HDF5 or similar technologies when widespread support is available.

7.1.2. Computational Limitations

There are two major computing components of the project that place limitations on the design. The input data for this project, as discussed in Subsections 3.3.2.1 and 7.1.1 above is the SDSS Catalogue, and it was decided that the results of this project should be output in the FITS file format. The constraints imposed by this choice of data, including issues that must be resolved by the software solution are discussed in Subsection 7.1.2.1.

The specific hardware implementation available to this project was Grid Ireland. As discussed in Subsection 7.1.2.2, that hardware solution imposed some constraints on the design of the project, and required specific elements of the software solution to be designed to accommodate grid computing.

7.1.2.1 Constraints Due to Input & Output Data

The input data for this project is the SDSS Catalogue of Calibrated Objects (usually referred to in this project as the “SDSS Catalogue” for brevity.) The entire catalogue was downloaded from the SDSS DAS using the `wget` utility in accordance with SDSS instructions. [156] The catalogue is stored in FITS data table files, which must be accessed using the FITSIO library. As discussed in Subsection 3.3.2.2, the language with the best access to that library during the development phase of this project was C, and it is used throughout this project.

The SDSS Calibrated Objects files (`tsObj` files) are defined to contain 146 columns [121], and have a measured mean of 847 rows, each consisting of a single observation of an object by an SDSS camera. Many of these objects are non-stellar sources, and many more are not the “primary” observation of a particular source as defined by the SDSS Image Processing Flags. [157] In addition, of the 146 columns, only three are needed for the Locus Algorithm as defined in Section 5.2: the two positional parameters, RA and Dec, and the model magnitude (itself a vector of five values: one each for u , g , r , i , and z [158]).

The input SDSS data sets are too large for frequent data transfer using the Grid hardware implementation. A necessary first step, therefore, is to reduce the data volume at each processing step. This has the additional benefit of providing for extensibility to other catalogues in future projects. To achieve this, an API, discussed in Subsection 7.2.1, has been developed to create a Local Catalogue consisting only of entries for a “clean sample of stars” (as defined by SDSS [157]) and with only the three necessary columns retained.

The files in this Local Catalogue are then used as the input for the data analysis pipeline defined in Subsection 7.2.2. This pipeline, iterated over each of the ~86,000,000 stars in the Local Catalogue was used to produce the output data for the Exoplanet Catalogue as discussed in Subsection 11.4.

The API and the Pipeline are programs written in C. In order to enable the programs to be developed in one environment and run in another, portability has been built into the design of the software. Input parameters to the program such as target lists and the

paths to catalogue files to be processed are stored in parameter files as defined in Subsection 8.3.3. These parameter files are generated by parameterisation software, described in Subsection 7.2.3. This software was developed in a modular manner to allow it to be modified to access the different file systems used in the project. Section 8.1 describes how the interactions between these file systems operate.

As shown in Subsection 5.2.1, for each target, it is necessary to identify the SDSS fields which can be included in a FoV with the given target as shown in Figure 5-4. Each of these fields, as discussed in Subsection 3.3.2.1, can be identified by the combination of four field descriptors: `run`, `rerun`, `camcol` and `field`. Each field then corresponds to a particular file in the SDSS Catalogue, whose filename and path are constructed using those four elements.

SQL queries to the SDSS CAS were developed by Dr. Eugene Hickey of the SCG at ITTD, as discussed in Subsection 7.2.5, and are used to identify these sets of field descriptors. [159] These queries return CSV files containing the descriptors as shown in Subsection 8.3.5 which are used as input to the parameterisation software.

7.1.2.2 Constraints Associated with the Hardware Implementation

As discussed in Section 3.1, the time required to process the SDSS Catalogue on a single computer was initially assessed at between three and eleven years. A HPC solution was therefore needed to permit the execution of the software solution as discussed in Section 3.2. The particular paradigm of HPC chosen was grid computing, and the grid which was used was Grid Ireland, as discussed in Subsection 3.2.1.

This choice imposes two constraints on the design of the project: Job Submission and Data Access. Firstly, using an EGI grid requires that the overall computing tasks (API and Pipeline) be divided into a number of grid jobs. Each grid job is specified by a “job file” written in a scripting language called the Job Description Language (JDL) as discussed in Subsection 3.3.2.3. Each JDL file, as described in Subsection 8.3.6, is a structured text file which specifies a single executable file and optionally specifies a number of parameters including node timeout limits, a set of data files to be uploaded to the node and a string which can be passed as a command line argument to that executable file [7]

These jobs are then assigned to worker nodes (WN) by the Job Submission System (JSS) as shown in Subsection 3.2.1. [7] When a job is submitted to the grid, the executable file is uploaded from gridUI to be run on the WN designated for the job, and the parameters and arguments are passed to it. Variations in these arguments, contained within the JDL file, are the means by which the overall tasks are divided into jobs.

Consequently, a different JDL file is required for each distinct job. [7] To complete the overall tasks, therefore, necessitates the creation of many such file. This, in turn, requires an automated process that creates unique JDL files from a template based upon input data. The creation and submission of these jobs, and the process by which they are tracked and monitored, is the basis of the grid management software suite, as discussed in Subsection 7.2.4.

A second constraint upon the design is imposed by the mechanism by which WNs actually access data. In particular, WNs do not have direct access to data stored in the LFC, but instead must access the LFC using `gLite` commands as discussed in Subsection 3.3.2.3. [7] However, because the C programs which are required to access FITS files do not allow for easy incorporation of these `gLite` commands, it is necessary to use BASH shell scripts as discussed in Subsection 3.3.2.4 as a wrapper. These BASH scripts call `gLite` commands to copy the required data and programs to the WN, execute those programs, and copy the output back to the LFC for storage as illustrated by Figure 3-3. These shell scripts are used as the executables in grid jobs.

7.1.3. Consequent Software Design

As suggested by Jalote, and as discussed in Subsection 6.1.1, constraints on a project can lend a Bottom-Up flavour to the design of the project. The constraints specified in Subsections 7.1.1 and 7.1.2 make certain elements of software mandatory in this project, and have a consequent influence on the design of other components of the software. The influence of this on the overall design is shown in Figure 7-1 and described below.

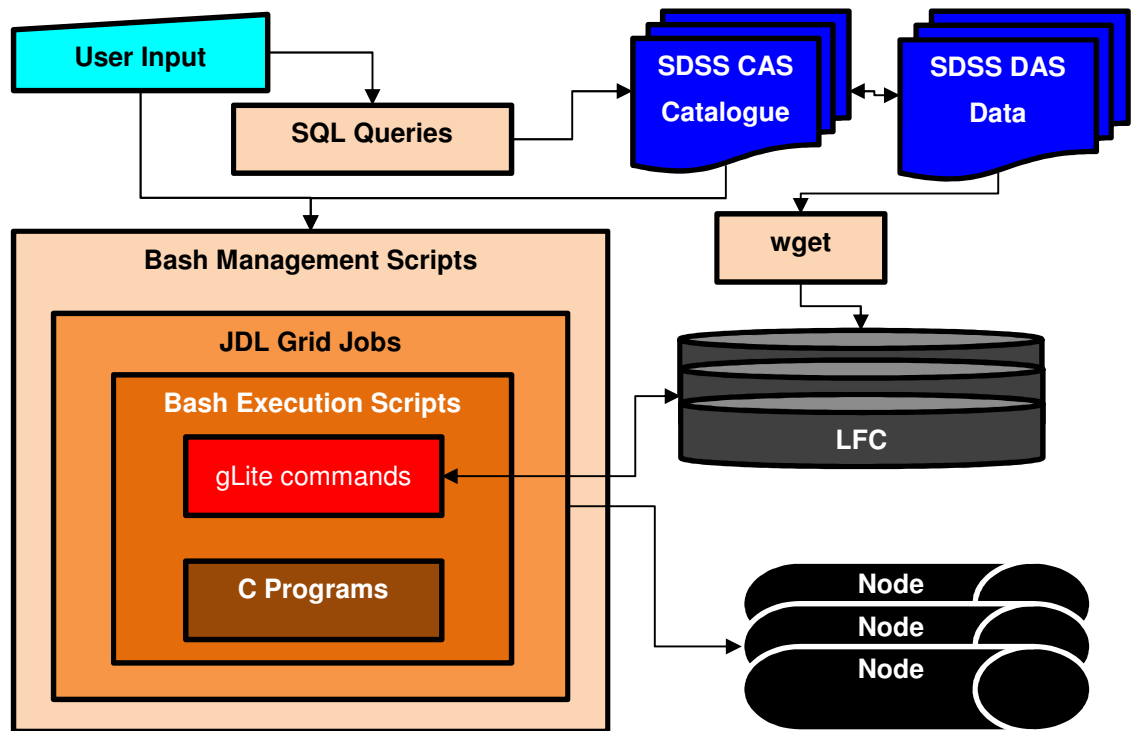


Figure 7-1: Bottom-Up Perspective on the Software Design of this Project

The major data processing operations, both in the API and Pipeline phases of the project, have to be carried out by a series of C programs, to enable access to the CFITSIO Library. These programs are executed on worker nodes within the grid environment. The input data for these programs is therefore required to be copied from the LFC to the WN, mandating the use of `gLite` commands. These commands are also used to copy the output from the programs to the LFC for storage.

Bash shell scripts are used as wrappers to combine the C programs and `gLite` commands into an executable suitable for use in a grid job. These jobs are specified by JDL files, which are, themselves, automatically generated, submitted and managed by grid management scripts, also written in Bash.

The input to the management software comes from direct user input and the outputs from the CAS. The user input specifies parameters including both job parameters (such as job size, maximum job number and so on) and astronomical parameters (such as telescope FoV, SDSS colour, and limiting colour and magnitude variations.)

The data from the SDSS CAS is generated by SQL scripts, which identify the corresponding SDSS DAS files needed for a set of targets. The contents of the SDSS

Catalogue are downloaded to the LFC using the `wget` utility through the SDSS DAS. This makes them accessible to grid jobs using `gLite` commands.

As is the nature of Bottom-Up design components, this set of constraints sets requirements that the software solution must include, but does not specify the exact manner in which they are to be implemented. Section 7.2 addresses the specific design of the project from a Top-Down perspective with reference, where appropriate, to these Bottom-Up constraints.

7.2. Overall Software Design

This Section uses the Top-Down, depth-first manner as defined in Subsection 6.1.1 to describe the design of the software solution of the project as it was implemented. The overall design is shown in Figure 7-2. It shows the high-level components which make up the overall design, each of which are discussed briefly below, and further decomposed in their own Subsections. The software developed for this project is intended for extensible use, potentially accepting source data from multiple catalogues as discussed in depth in Subsection 14.1.5.

The Application Programming Interface (API), discussed in Subsection 7.2.1 is designed to take the original source data (in this case, SDSS data), examine it to locate the required data for the main software processes, and extract only that data which is needed into a “tidy data” format as described by Wickham, 2014. [160] This format is known as the Local Catalogue, described in detail in Section 8.3.1.2, and is structured in a manner that can be used by the main data pipeline.

It is planned that the project will be extended to take in input data from another source, as proposed in Subsection 14.1.5. This will require that a corresponding API will be generated to suit, using existing SDSS API as a template.

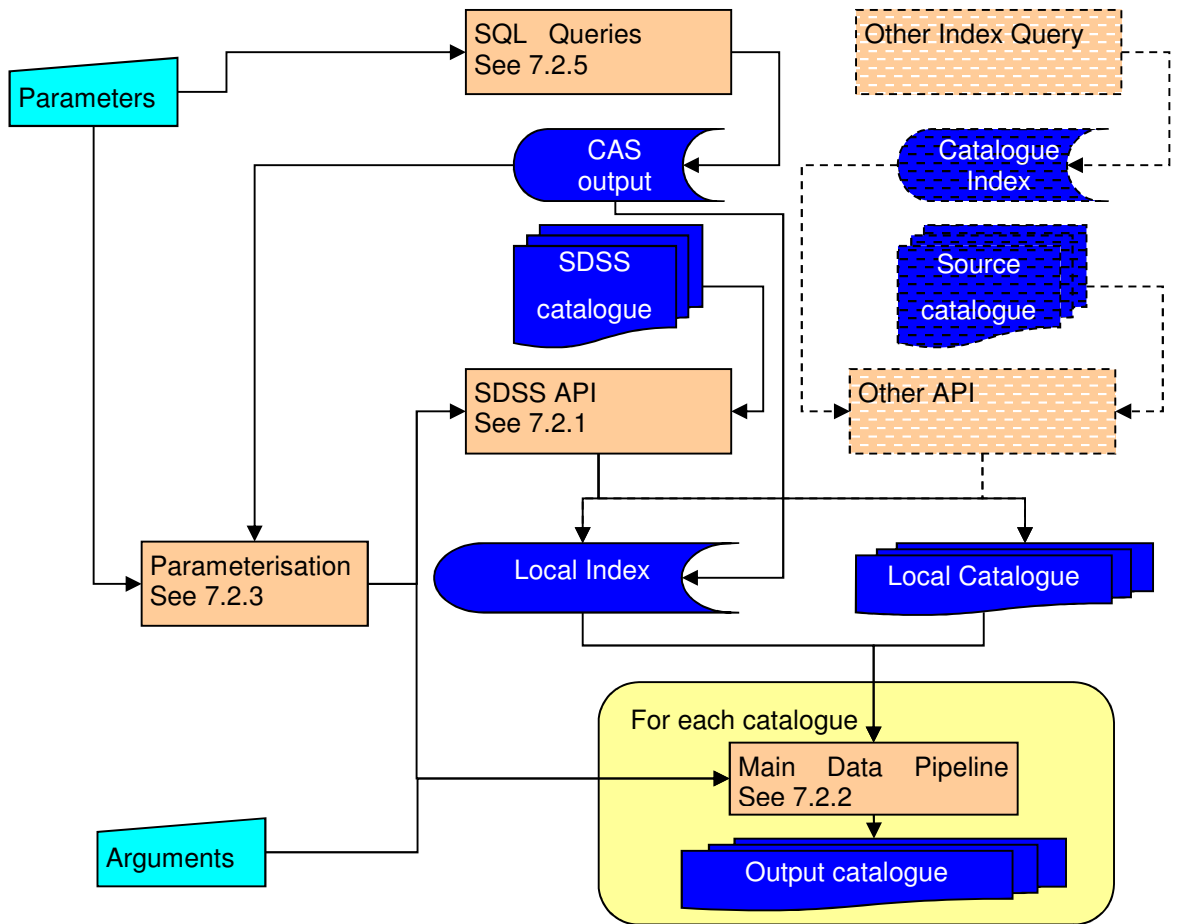


Figure 7-2: Top-Down Software Design: User Parameters define SQL queries to the CAS and are used by the Parameterisation system to define API jobs which process the SDSS catalogue to generate the Local Catalogue. The Local Catalogue is then processed with further user arguments in the Main Data Pipeline to produce the Output Catalogues. Each component is discussed in its respective subsection. Not shown are Grid Management (7.2.4) and Error Checking (7.2.6) systems which are integrated within other software elements.

The main data pipeline is a flexible process that uses parameter files and command line arguments to identify input data from the local catalogue and specify the particular analyses to be performed. By modifying the input parameters, the software can be run in two different modes (Catalogue Traversal and Target List), as discussed in Section 7.2.2. This project incorporates two primary analyses which used these modes: an analysis of quasars contained in SDSS and a full analysis of all stars in SDSS for their suitability for exoplanet search by the transit method. (See Subsections 4.1.2.1 & 4.1.2.2)

The Parameterisation program, discussed in Subsection 7.2.3, generates Parameter files for both API and Pipeline. Parameter files contain lists of the paths to input files needed for the API, and both files and targets for the Pipeline as discussed in Subsection 8.3.3. This program allows for flexible use of the two main components.

Grid computing allows each of these components (API, Pipeline and Parameterisation) to be run many times in parallel to deal with the volume of data that has to be processed. Each has their own suite of grid management software to generate, submit and manage grid jobs. As the various sets of grid management software have similar requirements, and thus are designed in a similar way, they are discussed together in Subsection 7.2.4.

Specific to SDSS is the requirement to identify fields by means of SQL queries to the CAS. These queries, described in Subsection 7.2.5, accept user parameters and generate lists of targets and field identifiers used as input to the parameterisation software. These queries are used in relation to both the Source and Local Catalogues because they use the same directory structure.

Within each of the C programs, a robust system of Error Checking is used to ensure reliable operation, and the identification of both fatal errors and warnings within the program. This system was developed as a single portable module and is used universally throughout the project. As such it is discussed separately in Subsection 7.2.6. Note that while error checking is not explicitly referenced in the definition of the programs below for clarity, all function calls and any other operations which could potentially raise an exception are subject to error checking.

7.2.1. SDSS Data Access API

The SDSS Data Access API is designed to extract data from the SDSS catalogue in such a way that the files used in the pipeline are minimised in size. Each instance of the API is executed as a grid job, submitted from gridUI by the grid management software in a JDL file, which calls a bash shell script called `call_api`.

`call_api` operates on a worker node, and takes two parameter files as inputs from gridUI. One is in PRT text format and one is in PRM binary format which are generated as part of the parameterisation process. The PRT file (see Subsection 8.3.3.3) specifies

the full paths to a set of input SDSS fits files which are to be processed by the API to create the corresponding local catalogue files and is used by the shell script. A *for* loop iterates through this file, and uses the `lcg-cp` command to copy the listed files to the WN.

Two programs, Diagnose and Extract, are then copied from the LFC to the WN. These programs use the PRM parameter file (see Subsection 8.3.3.1) as an input to access the fits files. The `std.out` from `diagnose` and `extract` are discarded to `/dev/null`, as the GMS has limited capacity for storing output. [134] `call_API` can operate in verbose mode for debugging, whereby this output is not discarded.

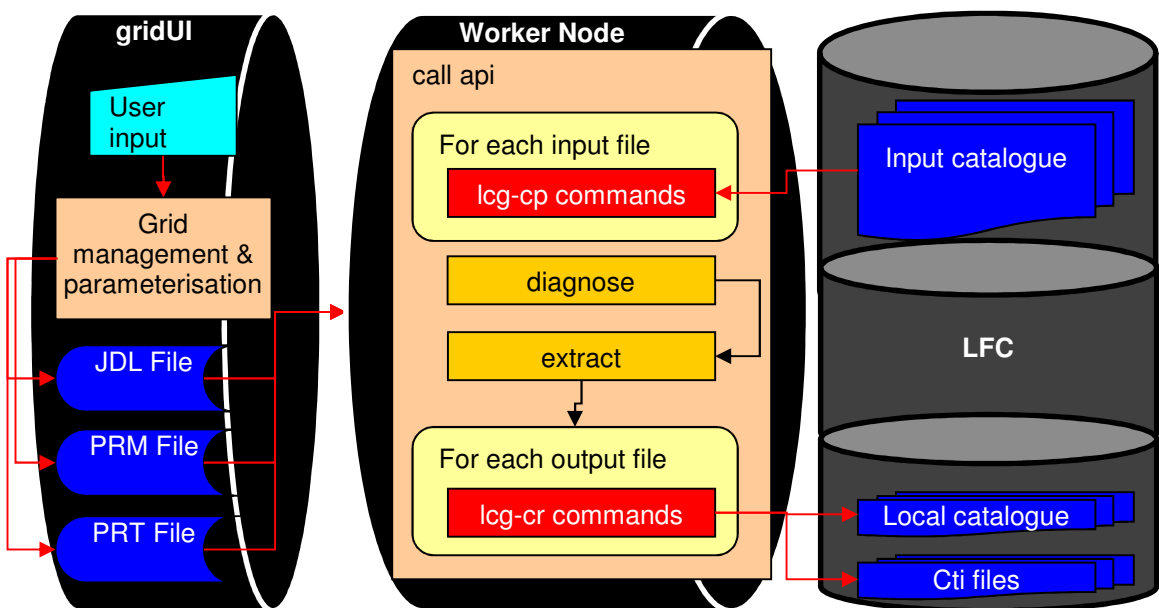


Figure 7-3: Data Access API

The Diagnose program (defined in Subsection 7.2.1.1) accesses and analyses the data contained within the input files, and identifies the columns in the data table. The column names together with information regarding their structure are then stored in a CTI file.

Next, the Extract program (described in Subsection 7.2.1.2) carries out the substantive work of the API. Taking the columns identified by Diagnose, it identifies the columns containing only the relevant data: Right Ascension, Declination and Magnitude (itself an array of 5 double values.) It then applies a reduced version of the SDSS clean sample of stars algorithm to exclude entries which are not primary entries for stars. The

data for the remaining entries in those three relevant columns for this project are then copied to a file which forms part of the local catalogue.

Finally, `call_api` then uses the `lcf-cr` command to copy all of the local catalogue files to the LFC for storage, and registers them to the catalogue.

7.2.1.1 Diagnose

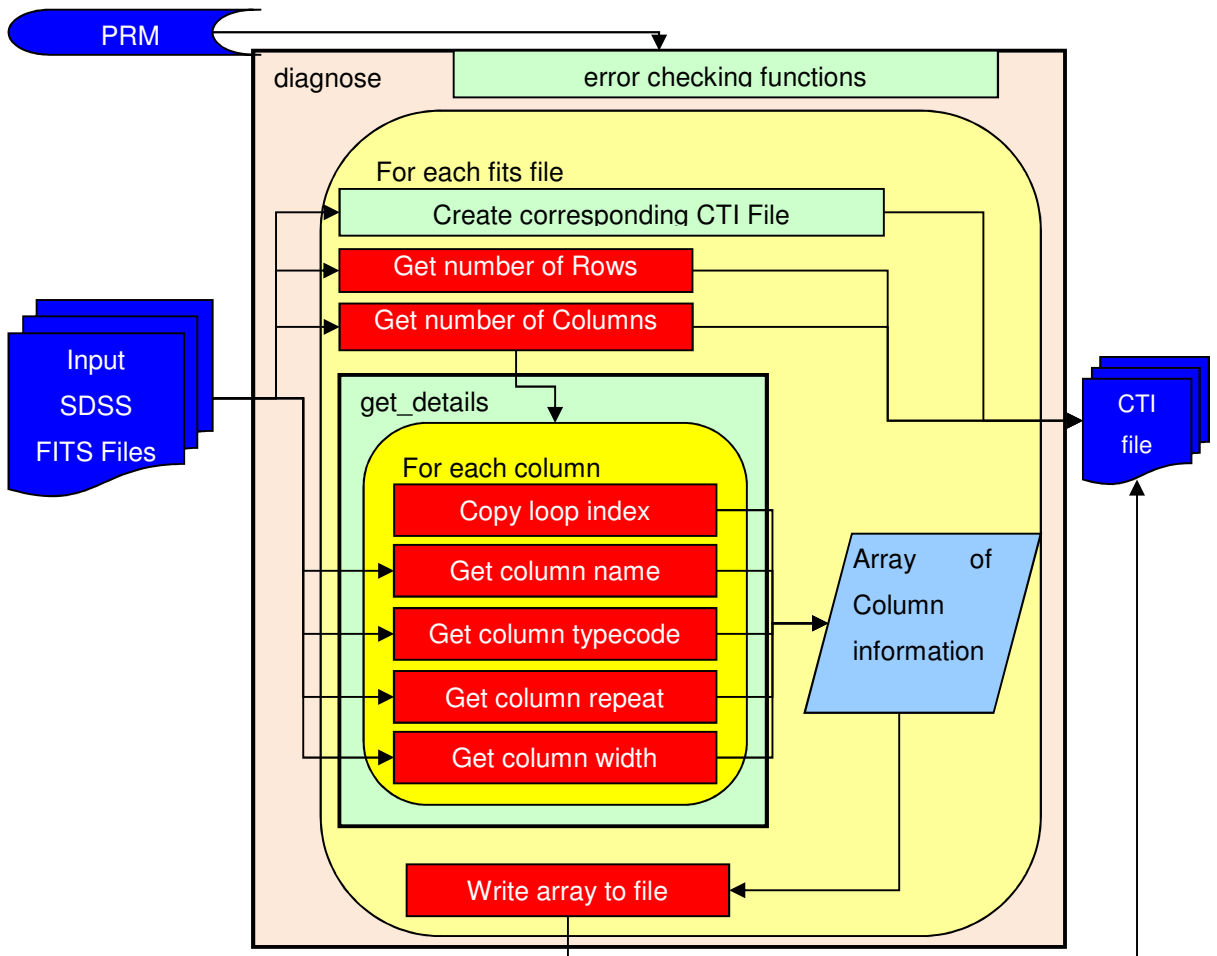


Figure 7-4: Diagnose

The Diagnose program is designed to examine fits table files, specifically those supplied by SDSS, and find out what data is in each column of the fits table. This information is then stored in CTI files (See Section 8.3.2).

The Diagnose program takes as an argument the path to a PRM file (see Subsection 8.3.3.1) and uses the standard suite of error checking functions, defined in Subsection 7.2.6, to ensure the input is correct. The PRM file includes a count of and lists the input

SDSS files to be processed by the program. The count is used to control a *for* loop that iterates over the list of fits files.

Within that primary loop, it accesses the fits files one at a time and creates a CTI file for each one with the same name with the extension changed from `.fit` to `.cti`. The program then determines the number of rows and columns in that file. For each column, it uses the loop index to create a column index, and *gets* the name of each column and its data type (`typecode`, `repeat` and `width`), and stores all of this information in an array. The number of rows, columns and the array of information on each column are then written into the CTI file.

7.2.1.2 Extract

Extract is designed to process a list of SDSS input fits files, and extract a subset of the data stored in each of those files to create the Local Catalogue files. This subset is defined to exclude any entries which do not meet a reduced version of the SDSS clean sample of point sources criteria [157], and consists only of three columns of data: the two position co-ordinates, RA and Dec, both stored as double precision floating point numbers, and the magnitude of the object: a vector of 5 double-precision values, one for each of the 5 SDSS colour bands (*u*, *g*, *r*, *i* & *z*).

Extract takes as an argument the path to a PRM file which contains a count of, and list of FITS file paths. For each FITS file, the program uses functions to generate the paths to the corresponding CTI file and the local catalogue fits file to be generated by modifying the string containing the path by replacing the `.fit` extension with `.cti` and `_local.fit` respectively.

The newly generated path to the CTI file created by the diagnose program is used to access that file by the `cti_extract` function. The data from the CTI file is stored in an array. The columns in which RA, Dec and Model magnitude are identified and passed to the `fits_extract` function.

The `fits_extract` function loops through each object in the source FITS file and applies a reduced version of the SDSS Clean Sample of Point Sources filter. This filter checks if a set of bit-flags contained within the columns `status`, `objc_flags` and

`objc_flags2` are set as shown in Table 7-1 and if the `objc_type` is `star`. The effects of this reduced version are discussed in Subsection 11.2.

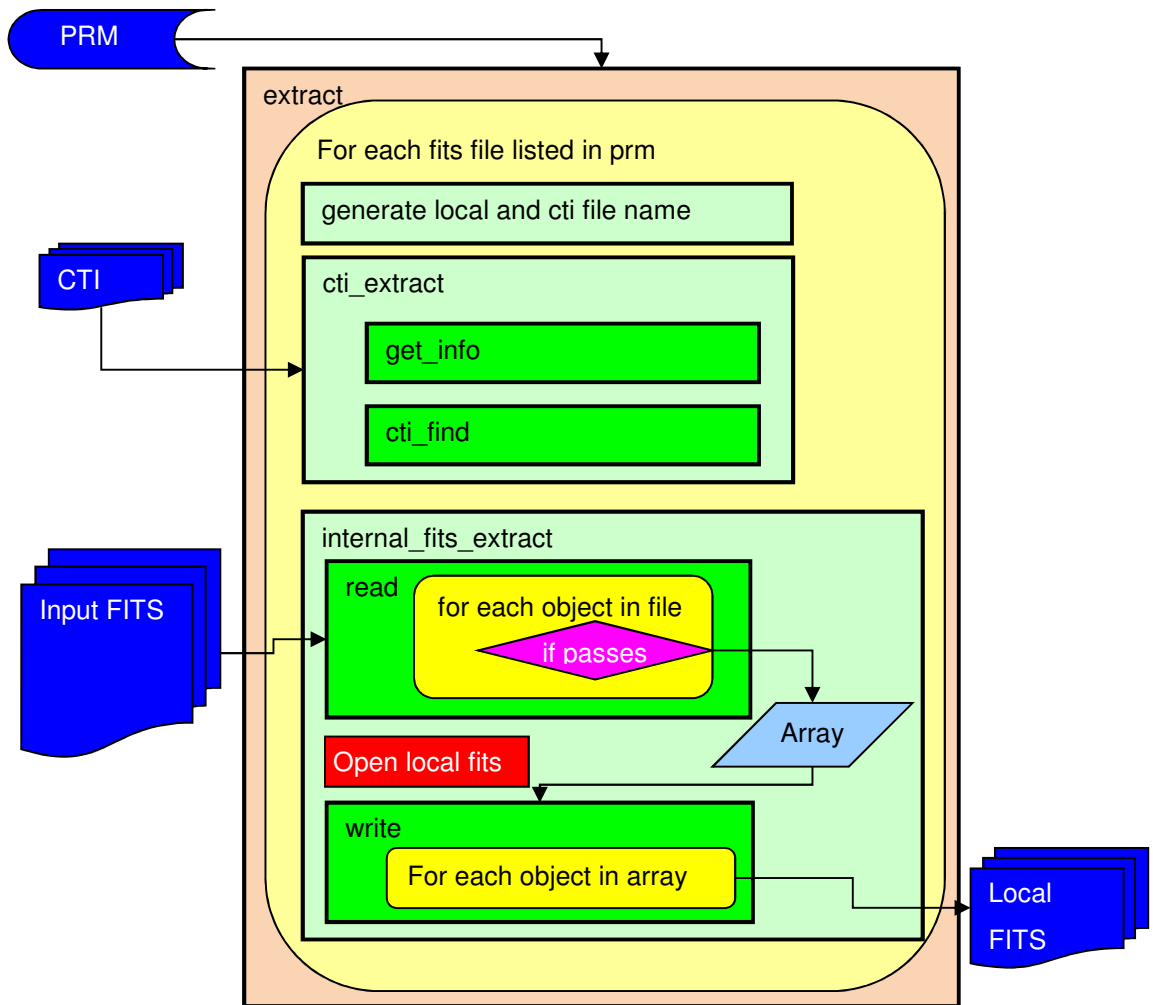


Figure 7-5: Extract

Objects that pass this filter are counted, and written to an array of structures which is passed to the `fits_write` function. The `_local.fit` file path is used to create a FITS file to store the output, and a FITS data table is created within that file. The contents of the array are written to this Local Catalogue file and the main loop moved on to the next file listed in the PRM file.

These files taken together form the Local Catalogue, which uses the same file and directory structure as the SDSS Catalogue, as discussed in Subsection 8.2.5.1

Column	Flag	Bitposition	Required Value
status	PRIMARY	13	TRUE
objc_flags	BINNED1	28	TRUE
objc_flags	PEAKCENTER	5	FALSE
objc_flags	NOTCHECKED	19	FALSE
objc_flags2	BADCOUNTSERROR	8	FALSE
objc_flags2	PSFFLUXINTERP	15	FALSE
objc_type	STAR	n/a	6

Table 7-1: Reduced version of SDSS Clean Sample of Point Sources Filter [157]

7.2.2. Data Pipeline

The Data Pipeline applies the Locus Algorithm as defined in Section 5.2 to a set of targets to produce output files which form the Output Catalogue(s). It operates in two modes: Target List and Catalogue Traversal.

Target list mode is used when a set of one or more targets are selected in advance by the user and are submitted to the pipeline for processing. This mode was used to produce the Quasar Catalogue as discussed in Subsection 4.1.2.1 and illustrated by Figure 7-6

Catalogue traversal mode uses an existing catalogue or subset of a catalogue to produce the target list. This target list is then submitted to the pipeline in a similar way to the Target list mode discussed above. This mode was used to produce the Exoplanet Catalogue as discussed in Subsection 4.1.2.2 and shown in Figure 7-7

Much of the software is designed to ignore the distinction between these two modes, for example, by treating a single target as a list of length one. Distinctions between operational modes are therefore discussed only as needed, as shown in Figure 7-8, where dashed boxes are used to indicate interchangeable modules.

In Target List mode, a target list consisting of the positions (RA and Dec) of a set of specific targets (e.g. quasars) is submitted through an SQL script to the CAS together with FoV size as discussed in Subsection 7.2.5. For each target in the list, the script identifies which fields are within the size of the FoV of that target as required by Subsection 5.2.1. The CAS returns a list of field identifiers (`run`, `rerun`, `camcol` and `field` as defined in Subsection 3.3.2.1) which can be used to generate file paths and names in the SDSS directory structure as discussed in Subsection 8.2.5.1. This list of identifiers is stored, along with the target data, in a CSV file, structured as shown in Subsection 8.3.5.2, which is saved on gridUI.

This CSV file is used, together with user input of observation parameters (FoV Size, resolution, maximum magnitude difference and maximum colour difference) as input for the parameterisation software. This software, as discussed in Subsection 7.2.3 creates a set of PPR and PRT parameter files. These files contain observational parameters, a list of targets and the paths to the files needed for each target. (For details see Subsection 8.3.3.2)

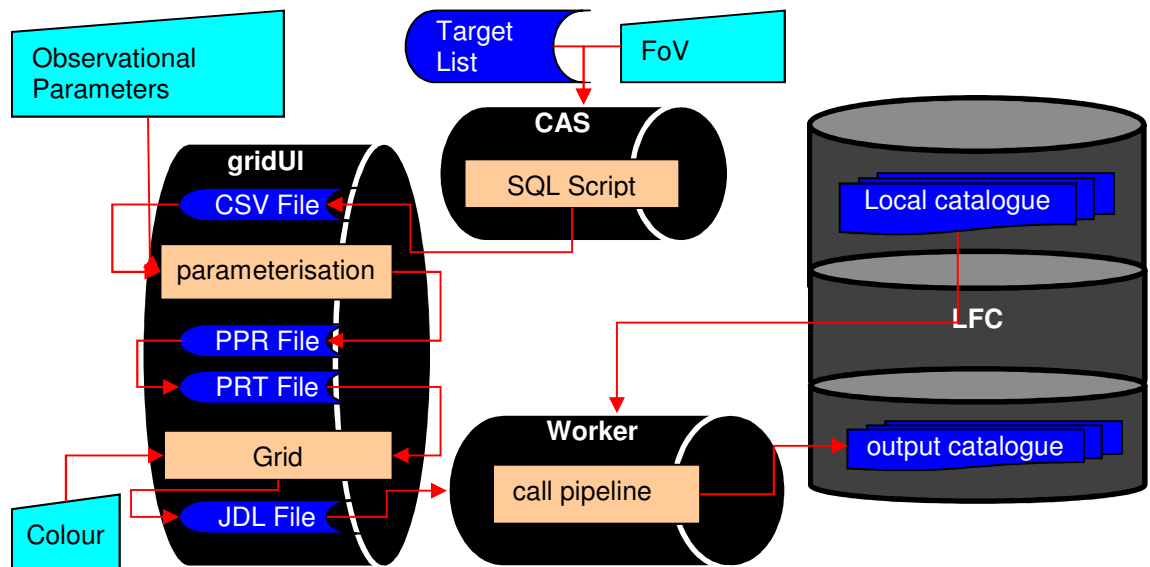


Figure 7-6: Operation of Pipeline in Target List mode

For each of these parameter files, a grid job is automatically created by the grid management software, which takes a command line input selecting SDSS band (stored as `colour`) and generates JDL files, then submits and monitors the grid jobs. The `colour` argument enables the same parameter files to be used for multiple SDSS bands. Each grid job calls a script named `call_pipeline` (discussed below) which uses the parameter files to identify the necessary local catalogue files and generates the output catalogue based on the listed targets.

In Catalogue Traversal mode, a field list, consisting of the field identifiers for all fields in the catalogue is submitted to the CAS instead of a target list. The SQL script identifies, for each target field, the neighbouring fields which have at least one object within the size of the FoV of any point in the target field. The set of identifiers for the target field are recorded with those of the neighbouring fields in a CSV file as shown in Subsection 8.3.5.3. This file is saved to gridUI.

The parameterisation process itself constitutes a grid-scale task in this mode. Grid management software divides the CSV file into smaller parts, which are each used to generate a JDL file for a grid job. This job runs the parameterisation software, which uses the fields listed in the CSV file to access the local catalogue file for that field. Data from the Local Catalogue is used in place of the target list data to generate the PPR and PRT files. This script then uploads these files to the LFC.

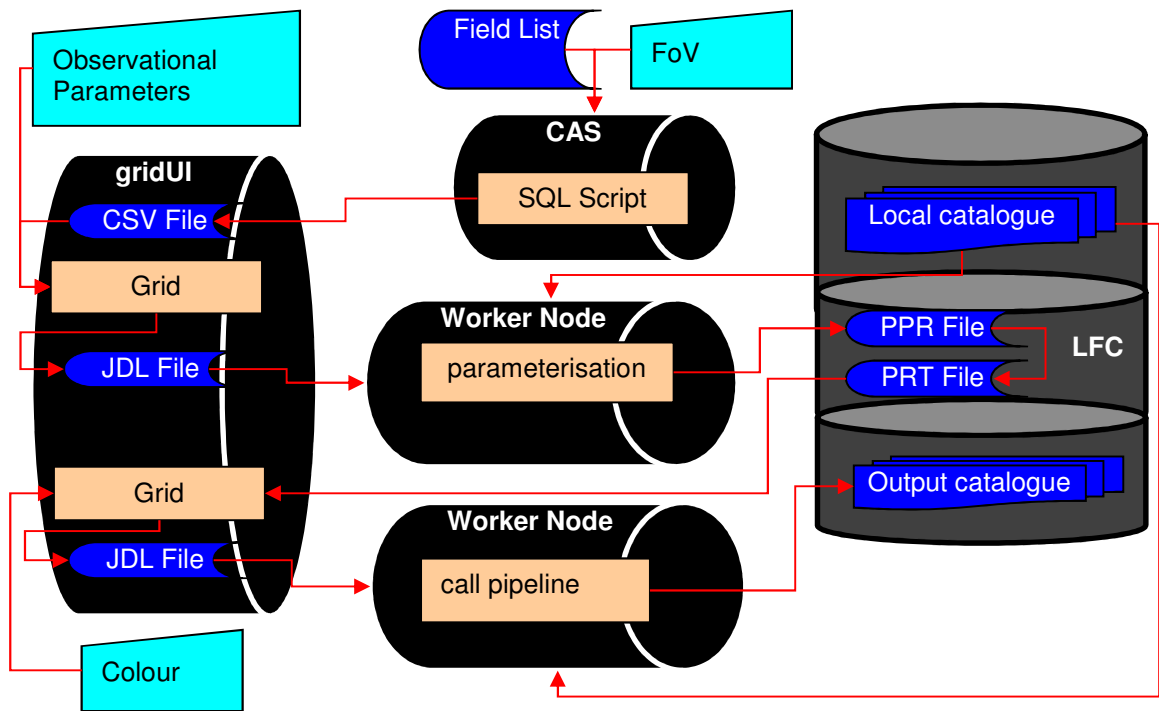


Figure 7-7: Operation of Pipeline in Catalogue Traversal mode

Grid Management software then creates a JDL file for each PPR file on the LFC and submits these to worker nodes. Each grid job accesses the PPR file on the LFC, instead of having it uploaded from gridUI as part of the job submission process, but is otherwise unmodified.

Two versions of the Call Pipeline script execute the pipeline, one in each of the two modes. The primary distinction between these modes is that when run in Target List mode, the Parameter files are stored on gridUI, while in Catalogue Traversal mode those files are stored in the LFC. This means that in the first case, the job submission process submits the PPR files as part of the grid job, while in the second, they have to

be copied out of the LFC using `gLite` commands. The two versions are otherwise identical as shown in Figure 7-8.

For each Local Catalogue file listed in the PRT file, `call_pipeline` checks to see if the file is already present on the WN, and if not, copies it from the LFC to the WN. Note that each file is likely to be used by more than one target in the target list.

Call pipeline then executes the C program `locus_algorithm`, defined in Subsection 7.2.2.1, with the colour argument provided in the JDL file, redirecting its output to `/dev/null` unless operating in verbose mode for debugging purposes. This program implements the Locus Algorithm and generates an output file.

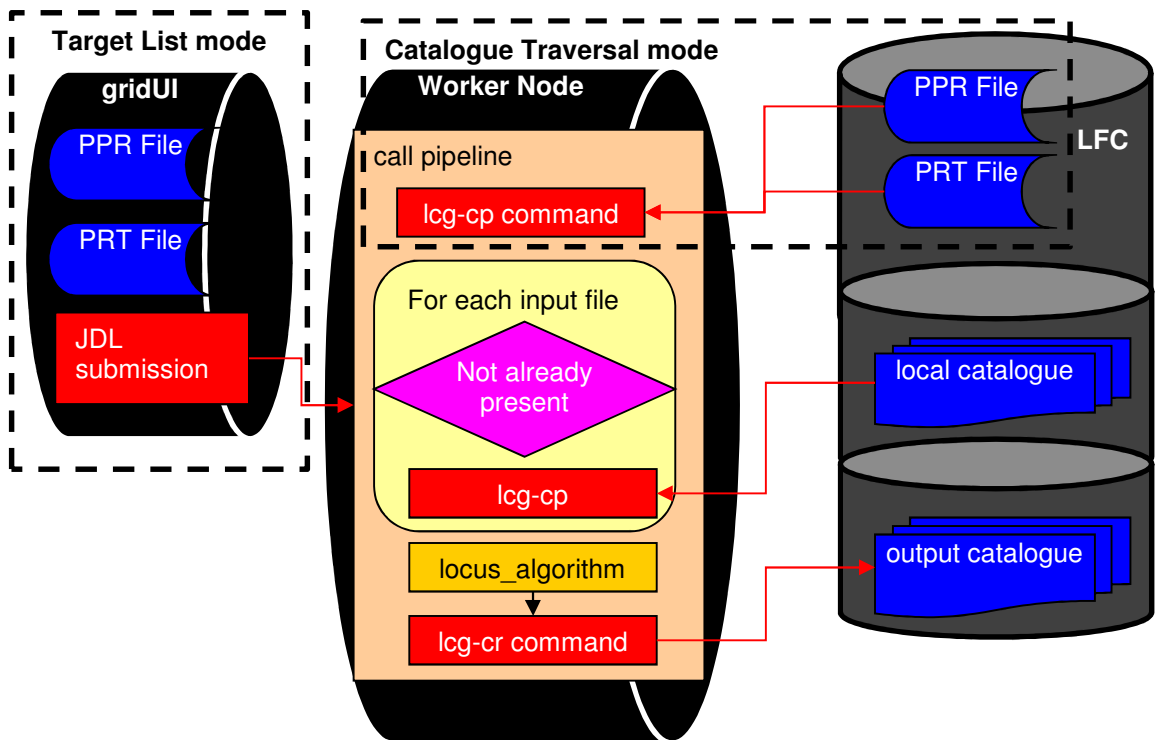


Figure 7-8: Call pipeline script (dashed boxes represent the modifications dependent on mode)

Finally, the output file is copied and registered to the LFC and any errors are reported to the Grid Management Software.

7.2.2.1 Locus Algorithm

The Locus Algorithm program implements the Locus Algorithm as specified in Chapter 5. This program takes in a PPR file and an SDSS colour as arguments, and requires

access to the Local Catalogue files listed in the PPR file. It analyses this input (regardless of pipeline mode) and produces output files which are combined to form the output catalogue

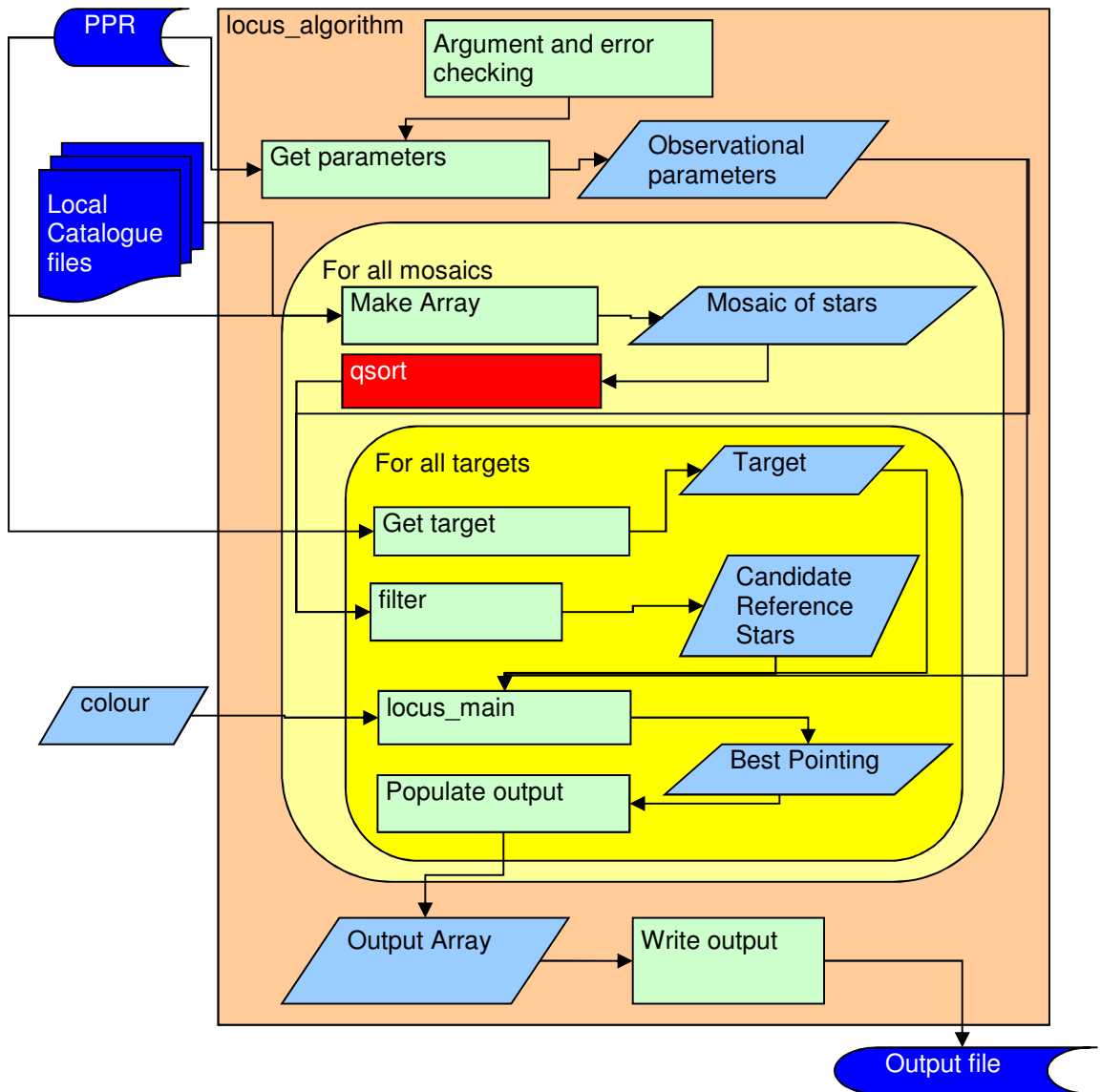


Figure 7-9: Locus Algorithm Program

As shown in Figure 7-9, after checking the arguments are valid, the program first retrieves the observational parameters from the PPR file and stores them in memory.

For each field listed in the PPR file, a mosaic of stars is generated to aggregate the data from multiple fields into a single array as required by the algorithm in Subsection 5.2.1. This array is then quick-sorted in ascending order of RA to allow subsequent functions to operate. Each star is then assigned a rating of 1 as an initialiser.

For each target listed in the PPR file corresponding to this mosaic, each of the steps in the bullet points below is carried out. (Note that in Target List mode, there is one such target per mosaic, in Catalogue Traversal mode there is a range of numbers of targets.)

- The `get_target` function stores the RA, Dec and Mag values for the target in a `struct` variable
- The `filter` function (discussed in Subsection 7.2.2.1.1) applies the filters of the Locus Algorithm to generate an array containing a short list of candidate reference stars as discussed in Subsection 5.2.2, and then assigns a rating to each candidate.
- The `locus_main` function (explained in Subsection 7.2.2.1.2) then applies the loci and identifies the best pointing for a given target. This pointing and its score are written to an output array.

Once all targets in all mosaics in the PPR file have been analysed, the output array is written to the output file and the program closes, reporting any errors using its exit code.

7.2.2.1.1 Filter

The Filter function generates a list of candidate reference stars for a given target. The Filter function takes as input the following data from the calling function:

- The array of stars created from the mosaic of input files, which have been sorted in ascending order of RA and initialised with ratings of 1
- Observational parameters (Resolution, FoV, maximum Magnitude difference and maximum Colour difference)
- Details (RA, Dec and mag) of the target
- SDSS colour number corresponding to the SDSS band ($u=0, g=1, r=2, i=3, z=4$)

For each star in the mosaic, the following steps are carried out to create an array of candidate reference stars as defined in Subsection 5.2.2. Note that all calculations related to RA are corrected for spherical effects as shown in Equation 5-2.

Firstly, the `sift` function is applied to check whether or not the star could be resolved and assigned it a rating of 0 if it could not. `sift` uses a loop to compare the star with each of the subsequent stars in the mosaic with the following rules

- `sift` terminates when it reaches the end of the mosaic
- `sift` terminates when the difference between the RA of the star and the subsequent star it is being compared against is greater than the resolution limit
- If the absolute difference between the Dec of the star and the Dec of the subsequent star is less than the resolution limit, then:
 - If one star is more than 5 magnitudes fainter than the other, then the fainter star is assigned a rating of 0
 - Otherwise both stars are assigned a rating of 0

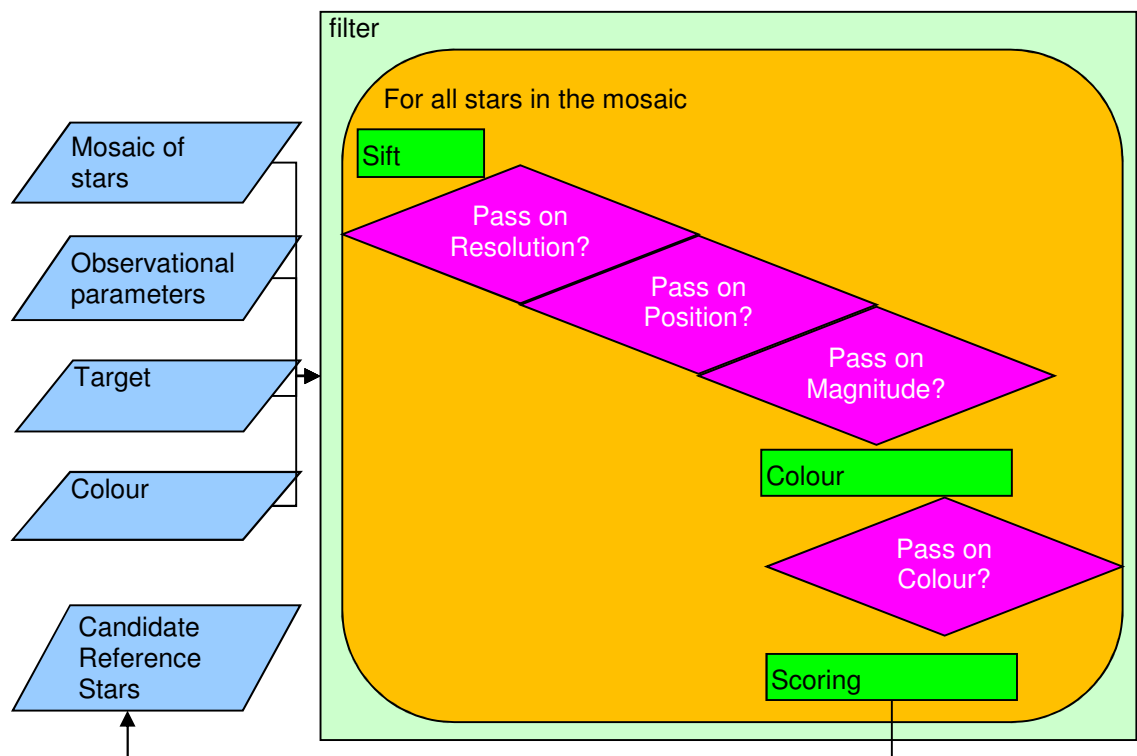


Figure 7-10: Filter Function

A sequence of nested *if* statements as listed below are used to test to see if a star in the mosaic is a viable candidate reference star.

- If a star currently has a rating of 1 (this means it passed the test in `sift`)
- If the absolute differences in RA and Dec are less than the FoV size
- If the absolute difference in Mag is less than the maximum magnitude difference

The function `colour_comparison` is called inside this nested set of conditions which have the following steps:

- It calculates the absolute difference in colour index between the selected filter and all the *neighbouring* bands (e.g. $u-g$ if u is selected, $g-r$ and $r-i$ if r is selected)
- If the difference in each of these indices is less than the maximum colour difference, then it passes the colour test, and a `colour_match` flag is set.

If all of the above tests are passed, the `scoring_mechanism` function is called

- If the SDSS colour is the first or last band (i.e. u or z), it uses the Triangular rating formula defined in Equation 5-4.
- For bands with two neighbouring bands (SDSS bands g , i , or r) it uses the mechanism for combining ratings from all neighbouring bands as defined in Equation 5-6.

This produces an array of candidate reference stars, each listed with their individual ratings which is passed to subsequent functions

7.2.2.1.2 Locus Main

The Locus Main function calculates the optimum pointing for a given target. The complete set of these pointings forms the basis of the output catalogues of the entire project. The function takes as input the target data, the array of candidate reference stars near that target, the observational parameters and SDSS colour number. It produces an entry in an array of targets and pointing. These entries each consist of the following

- Target position (RA, Dec)
- Target Magnitudes (5 bands)
- Pointing position (RA, Dec)
- Score for the pointing

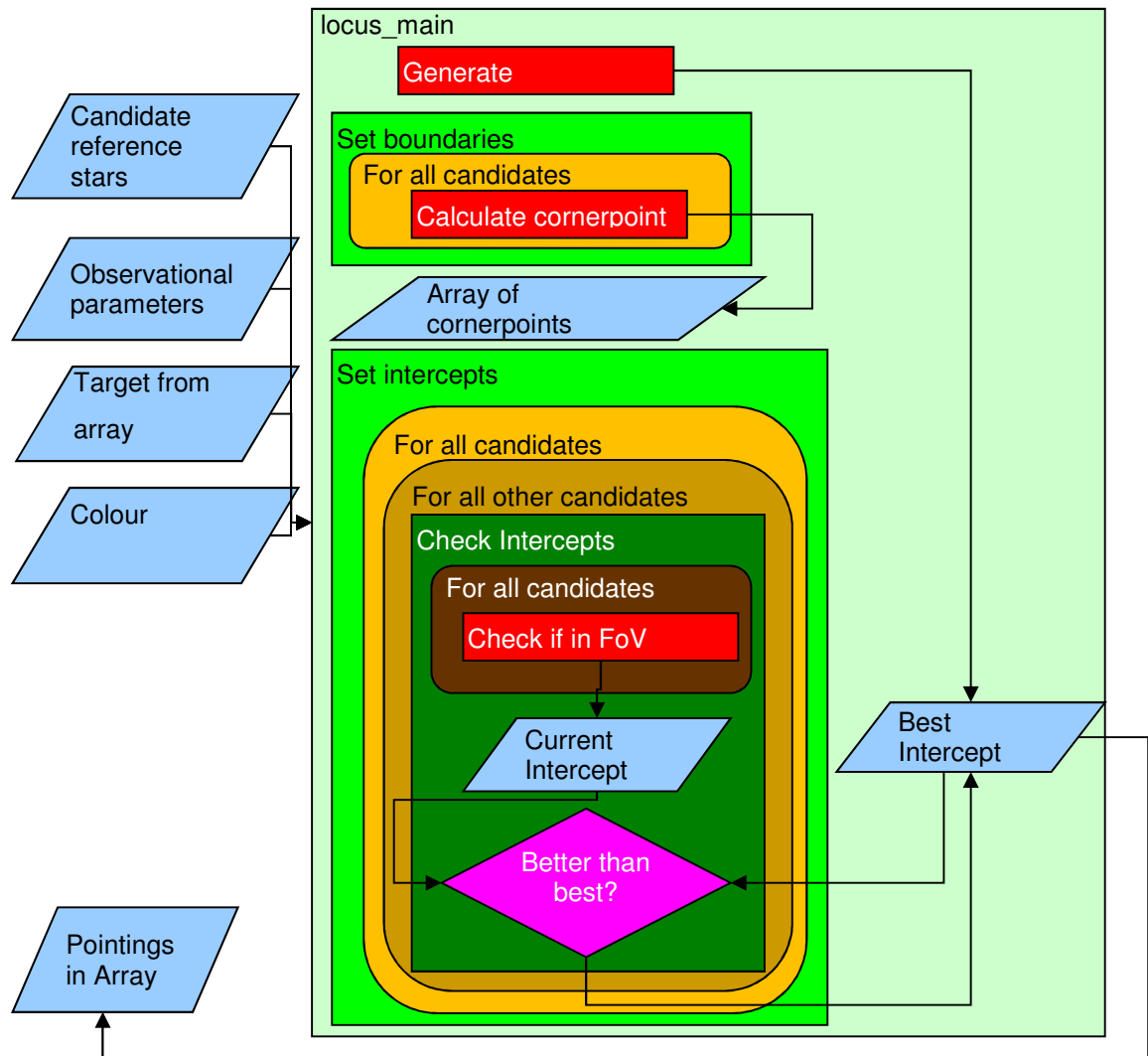


Figure 7-11: Locus Main Function

The Locus main function, as shown in Figure 7-11 has the following key steps

- The `locus_main` function first creates a placeholder variable `best_pointing`, with a value initialised to RA 0, Dec 0, and Score 0.
- The `set_boundaries` function then generates the loci for each candidate reference star by calculating the cornerpoint of that locus, and a bit-switch for its direction in accordance with Equation 5-3. These cornerpoints are stored in an

array with a one-to-one correspondence with the array of candidate reference stars

- The `set_intercepts` function then identifies the points of intersection between these loci in accordance with Subsection 5.2.4 by using the RA of one cornerpoint and the Dec of another.
- `check_intercepts` then checks, for each candidate reference star, if it is within half the size of the FoV of the intercept. It combines the ratings for each candidate reference star near that intercept into a score as described in Subsection 5.3.2. This pointing and score are stored as `current_pointing`
- Next, `current_pointing` is compared with `best_pointing`. If the score for `current_pointing` is higher than that for `best_pointing`, `best_pointing` is updated to the value of `current_pointing`.

Finally, the best pointing for a given target is output into an array which is used to generate the output file in accordance with Subsection 5.2.5 as shown in Figure 7-9. The collection of these output files form the output catalogues and the primary result of the project.

7.2.3. Parameterisation

The Parameterisation software is designed to provide flexible input to the API and Pipeline programs. Early prototyping on these programs revealed that they would become heavily dependent upon SDSS file and directory structures without a layer of data abstraction to allow this information to be passed in by means of a parameter file. The parameterisation software generates two types of binary parameter file, PRM files for the API, and PPR files for the Pipeline. These files contain explicit file paths, target lists and control variables for the API and Pipeline programs. These files are discussed in Subsection, 8.3.3.

When developing the bash shell scripts used for grid management, it was determined that these formats can not easily be interpreted by the shell. As a pragmatic solution to this problem, text files containing the file paths were implemented which are accessed by the shell scripts, as a supplement to the binary files used by the C programs. These

files, though stored with the file extension `.txt`, are referred to as Parameter Text (PRT) files in this Thesis to distinguish them from ordinary plain text files.

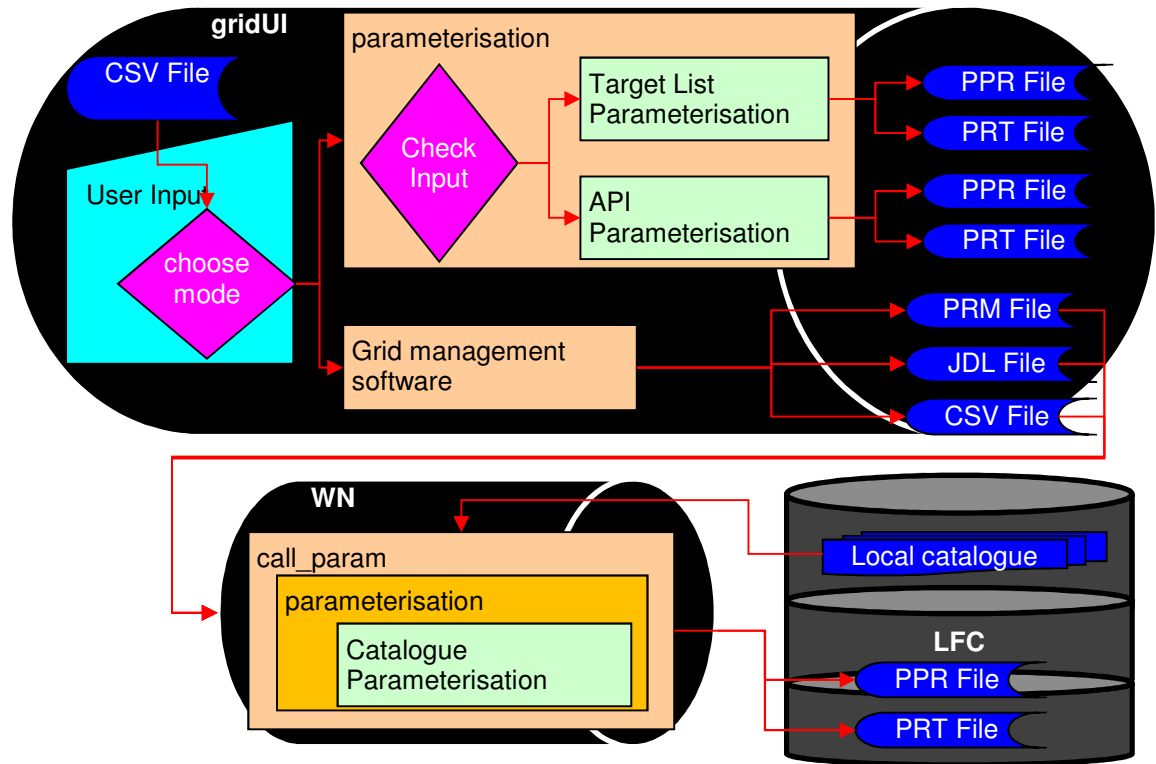


Figure 7-12: Parameterisation Modes. User input, together with the data contained in the CSV file from the CAS determines which mode parameterisation employs. Target List and API parameterisation run on and store their output on gridUI, while Catalogue Parameterisation mode is a grid job, managed by its own suite of GMS, storing the parameter files generated on the LFC.

The primary component of this system is a program called `parameterisation`. This program takes as input a combination of command line arguments and data stored in CSV files of various structures as described in Subsection 8.3.5. The output of this program is a number of parameter files, the size and structure of which is determined by command line arguments which designate which mode of operation the program employs.

`parameterisation` is a program that switches modes based upon the command line arguments with which it is called. It operates in three modes: API, Target List Pipeline and Catalogue Traversal Pipeline. In all three modes, input consists of user input arguments and a CSV file from the CAS. Different inputs are used to determine in which mode Parameterisation operates. Regardless of mode, `parameterisation` takes the following common arguments

- A number indicating the size of grid job to generate
- The path in which the output parameter files are to be stored.
- The path to the CSV file containing data to parameterise
- The path that FITS files identified by the parameter files are stored in.

Input for the API mode has no additional arguments and the CSV file, as discussed in Subsection 8.3.5.1, includes just the four columns used to identify fields: `run`, `rerun`, `camcol` and `field` as defined in Subsection 3.3.2.1. When `parameterise` is called with these four arguments, it operates in API mode as described in Subsection 7.2.3.1 below. .

Input for both pipeline modes (Target List and Catalogue Traversal) includes information about the targets to be processed in the pipeline. In Target List mode, this information is stored in the CSV file. This file includes information on each target and identifiers for the fields around that target, the details of which are described in Subsection 8.3.5.2. In addition, it is necessary to include observational parameters regarding the telescope the output is intended for use with. As a result the following arguments must be added by the user.

- Field of view size (in degrees)
- Telescope resolution (in degrees)
- Maximum magnitude difference (in magnitudes)
- Maximum colour index difference (in magnitudes)

When `parameterise` is called with these eight arguments, it operates in Target List Pipeline mode as discussed in Subsection 7.2.3.2.

In Catalogue Traversal Mode, the Parameterisation software generates parameter files which list all stars in a given catalogue as targets, together with the fields which are needed to create a mosaic for each group of targets.

The CSV file used as input here, as discussed in Subsection 8.3.5.3, includes a list of target fields, each listed together with the corresponding neighbouring fields needed to generate a mosaic around them. Both the target field and their neighbours are identified by their field descriptors as above. `Parameterise` has to access each of these target

fields, to generate a list of targets to be used with the mosaic of field as discussed in the description of the pipeline software in Subsection 7.2.2 above.

As a result, the scale of the Catalogue traversal mode is much greater than the other two modes, it has to be operated as a grid task in its own right. Therefore the CSV file generated as output from the CAS is subdivided by the grid management software into a number of smaller CSV files. These smaller CSV files are used as input to each of the parameterise grid jobs in turn.

This grid management software uses Parameterise in API mode to generate PRM files corresponding to each small CSV file. As in the API, these PRM files give the file paths to the fits files which are needed for the program. A grid job is generated for each PRM file. These jobs are then submitted to the grid where a script called `call_param` carries out the grid job.

`call_param` first downloads the parameterise program from the LFC. It then downloads the local catalogue files specified in the PRT files supplied to the WN. Next, it runs `parameterise` in catalogue traversal mode, generating PPR and PRT files. Lastly, these output files are copied to and registered on the LFC using `gLite` commands in `call_param`.

The `call_param` includes this PRM parameter file with the arguments to `parameterise`. With this set of nine parameters, the program operates in Catalogue traversal mode as described in Subsection 7.2.3.3.

7.2.3.1 API Parameterisation Software

In API mode, the parameterisation software generates PRM parameter files (defined in Subsection 8.3.3.1) and corresponding PRT text files. These files are used in the API and in the parameterisation process for the pipeline in Catalogue traversal mode. Each API or parameterisation grid job corresponds to one PRM file, and the number of files listed in the PRM determines job size.

The four arguments for the API mode are discussed in Subsection 7.2.3 above. In this mode, the number indicating the size of grid job refers to the number of fields, from those listed in the CSV file, to include in each of the PRM files produced.

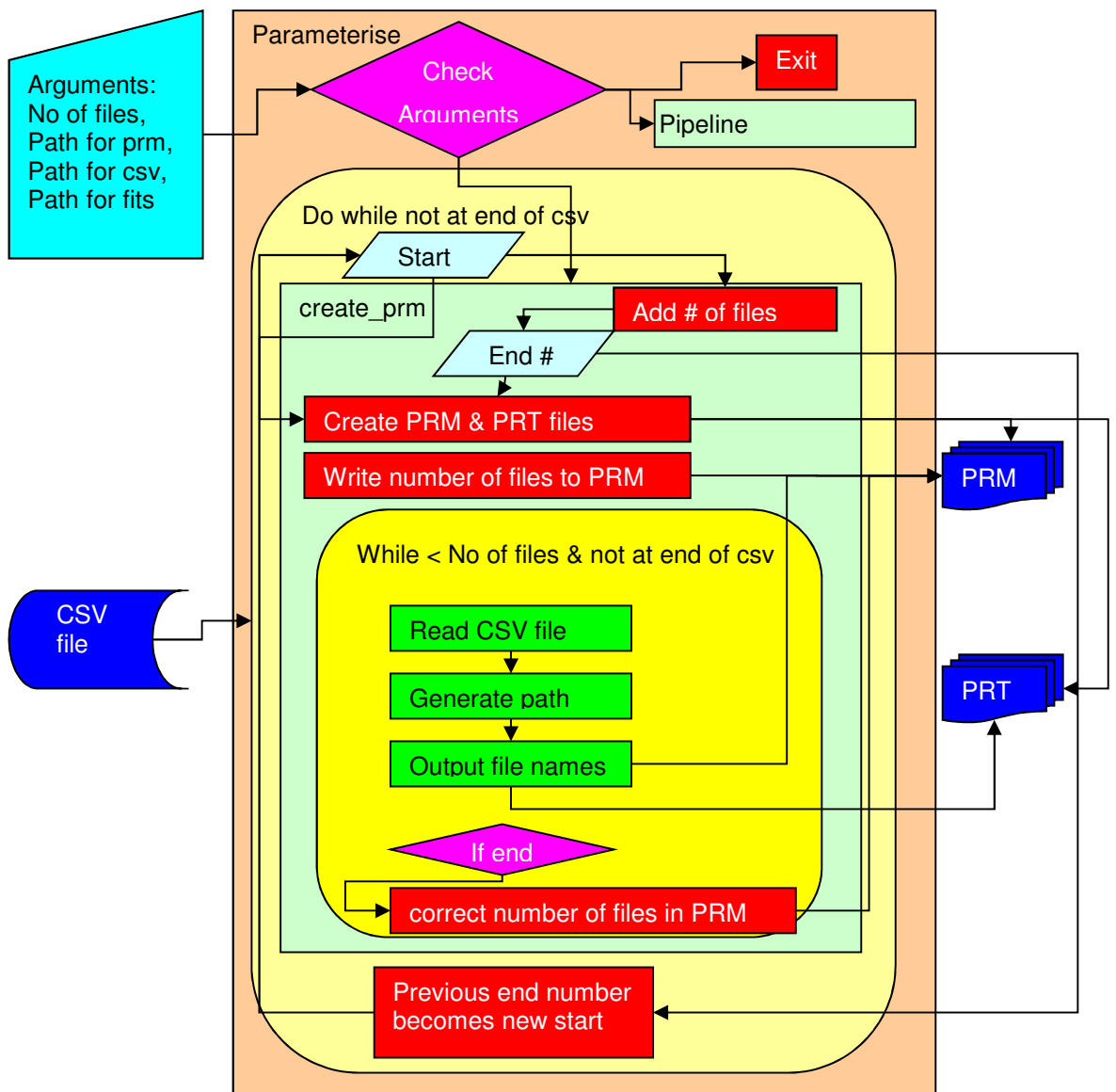


Figure 7-13:API Parameterisation

The first thing that the program does is to check its arguments to identify if it is being used to Parameterise for the API or Pipeline. If the arguments supplied by the user are unsuitable for either purpose, the program exits with an error code as discussed in Subsection 7.2.6.

Next, the `start_number` variable is initialised to 0 and the `end_number` variable is initialised to the number of fields per PRM. These variables are used to define the file names for the output files from parameterisation.

A *do-while* loop is then used to iterate over the CSV file from start to finish. When the end of the CSV file is reached, the program terminates.

Each iteration of the loop calls the `create_prm` function, which creates a new PRM and PRT file. The path into which the parameter files are written is an argument given by the user, but the individual filenames are determined at runtime. Each filename is the path given as the PRM path argument, appended (with underscore separators) with `start_number` and `end_number`, and the extension `.prm` or `.txt` as appropriate. For example, if `parameterisation` is run with a number argument of 50, and a PRM path `test/21-3-2012/parameter`, the third PRM file produced by the `parameterise` program would be `test/21-3-2012/parameter_100_150.prm`

Once the file is created, the number of fields per file is written to the PRM file as an integer. (Note that data in PRM files is written as binary data rather than text.)

Next, the CSV file is accessed in a *while* loop which iterates as long as its index is less than the number of fields, and the end of the CSV file has not been reached.

As described in Subsection 7.2.3 above, this CSV file has four columns each containing an integer number: `run`, `rerun`, `camcol` and `field`. These four numbers together identify a field in the SDSS catalogue. These numbers are used to generate both the file name and the directory path of each file in accordance with the SDSS data structure. A detailed description of how these paths are generated is given in Subsection 8.2.5.1. These paths are then written to the PRM and PRT files sequentially.

If the CSV file comes to an end before this loop reaches the number of fields, the program rewinds and edits the PRM file by writing the *current* loop index in place of the maximum number of fields. The program then closes all open files and terminates.

If this loop reaches the input number of fields before reaching the end of the CSV file, the parameter files are closed. The outer loop finishes by incrementing the Start Number and End number by the number of fields, and then iterates again.

7.2.3.2 Target List Pipeline Parameterisation Software

The Parameterisation software in both pipeline modes generates a series of PPR parameter files and their corresponding PRT text files, each of which describes a single grid job to the pipeline software. PPR Files, as defined in Subsection 8.3.3.2, contains

- observational parameters for a particular output catalogue
- a series of data blocks, each of which lists
 - The number of and the paths to a set of Local Catalogue files needed to create a mosaic suitable for use in the Locus Algorithm
 - The number of and details of the targets for which that mosaic is to be used

This Subsection describes how the parameterisation software operates in target list mode, however, only the target input module as highlighted in Figure 7-14 changes between this mode and catalogue traversal mode. As a result, most of this Subsection applies to both modes, and only the altered target input module is described in detail in Subsection 7.2.3.3.

In Target List mode, parameterise takes eight arguments as specified in Subsection 7.2.3. One of these is the path to a CSV file containing the details (RA, Dec, Mag) of a list of targets, each of which is associated with a set of field descriptors (run, rerun, camcol, field) for the fields needed to generate a mosaic about that target.

The `ppr_parameterise` function begins by creating a set of variables: `start_number` (initialised to zero) and `end_number` (set to the number of mosaics to be grouped together.) As shown in Figure 7-14, a *do-while* loop creates a new PPR file at each iteration, the paths to which are determined by `start_number` and `end_number`, and the given path in the same manner as in the API Parameterisation phase above. Next, it writes the observational parameters and the number of mosaics provided to the PPR file at that point.

The next loop reads in the field descriptors from the CSV file and uses the same methods as used in the API Parameterisation stage to generate a series of file paths for

the FITS files that are needed to make each mosaic. These file paths are written to both the PPR and the PRT files.

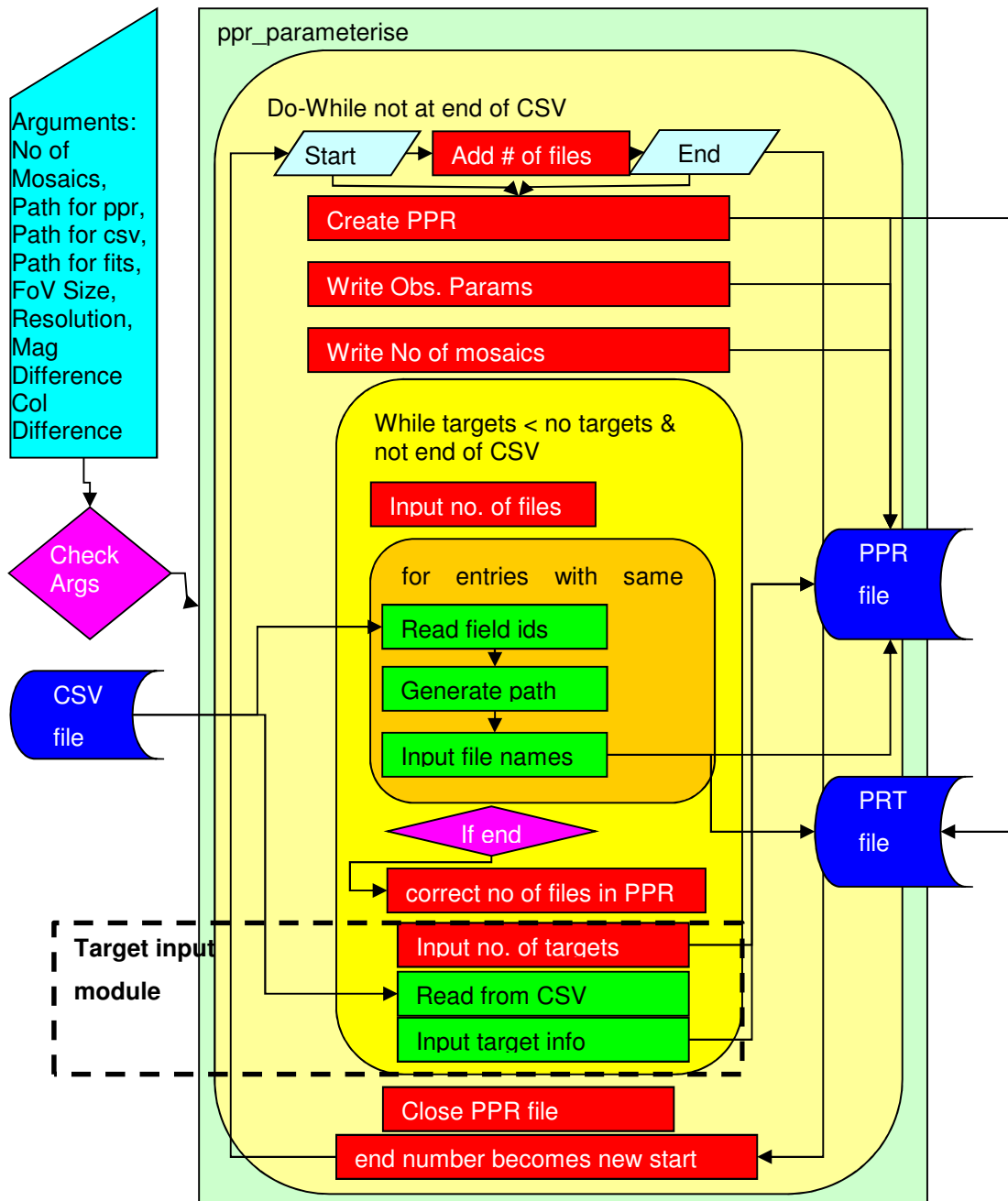


Figure 7-14: Pipeline Parameterisation. Note that the target input module shown here is the one for target list mode. Figure 7-15 shows the changes made for Catalogue Traversal Mode

The Target Input module is then called. In this mode, the program reads in the data about the target from the CSV file. Next, the integer 1 is written as the number of targets in the mosaic and the data on that target from the CSV file is written to the PRM

file. This completes the target input module which is replaced with a different module when operating in Catalogue Traversal mode as discussed in Subsection 7.2.3.3.

The final stage of the outermost loop closes the PPR and PRT files and incrementes `start_number` and `end_number` (which are used to name the files) by the number of mosaics.

Each loop also includes several checks to identify if the end of the CSV file had been reached early. When this happens, the program rewinds the PPR file and rewrites the number of mosaics contained in the PPR file to the correct count of mosaics. It then closes any open files and exits.

7.2.3.3 Catalogue Traversal Pipeline Parameterisation Software

In Catalogue Traversal Mode, the Parameterisation Software generates PPR files which describe mosaics centred on entire fields worth of targets, instead of single targets from a list as is the case in Target List mode.

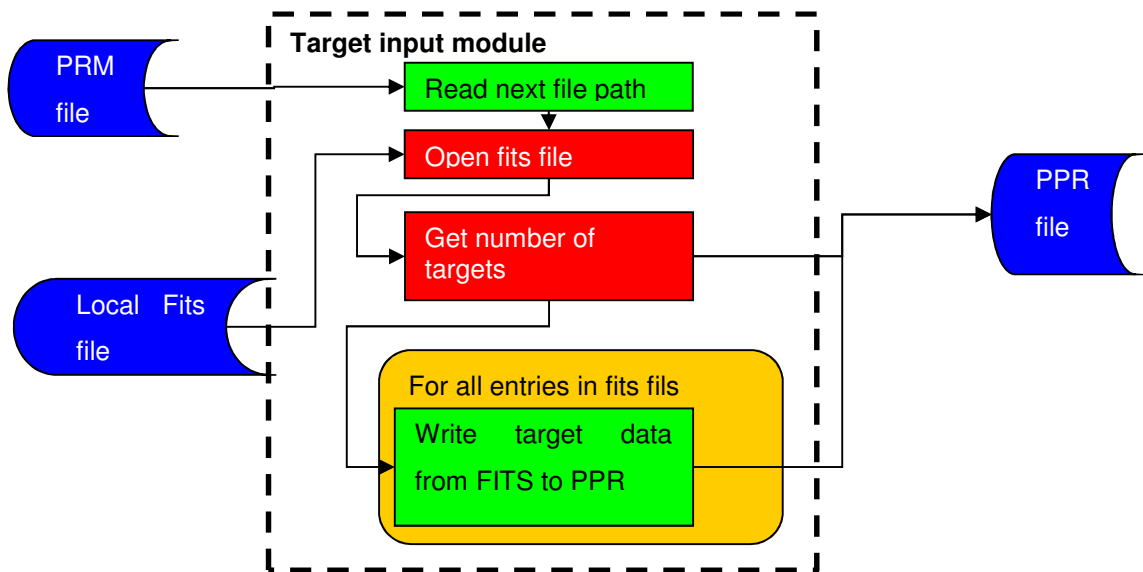


Figure 7-15: Target Input module for Pipeline Parameterisation in Catalogue Traversal mode. Note that as PRT files only include FITS paths, the target input module produces no output to the PRT file.

There are two primary differences in the design of the parameterisation software between these two modes. Firstly, traversing an entire catalogue and generating PPR files based on each field within that catalogue constitutes a grid job as discussed in

Subsection 7.2.3. Secondly, the Target Input module described in Subsection 7.2.3.2 above has to be replaced with one which could read Local Catalogue files specified in a PRM file as shown in Figure 7-15.

In this Target Input Module, the next entry in the PRM file is read, which contains the path to the fits file for the field around which the mosaic was to be generated. The module reads the number of entries in the FITS file, and writes that as the number of targets in the PPR file. It then loops through the FITS file, writing the details of the stars in that FITS file into the PPR file as targets.

7.2.4. Grid Management Software

This project used gris computing for four primary roles. All grid operations require grid management software to operate. The four modes are listed below.

- API
- Pipeline in Target List mode
- Pipeline in Catalogue Traversal mode
- Parameterisation for the Pipeline in Catalogue Traversal mode

For each of these roles, the same four components of the Grid Management Software (GMS) exist: The design of these components is largely identical across each of the four roles. The differences are treated here as minor variations to the design and highlighted where significant. The four components of the GMS are listed below.

- job generation
- job submission
- job calling
- job monitoring

The `generate_jobs` scripts operate on gridUI. They use existing Parameter files generated by the `parameterise` program in one of its modes to create a series of JDL files, each of which is submitted as a grid job. Parameter files are stored on gridUI except in the case of the Pipeline in Catalogue Traversal Mode, in which case the Parameter files are on the LFC. These files are listed (using `ls` or `lfc-ls` commands depending on the location of the parameter files) and the output of that list is piped by

script into a template using scripting variables. The output of these scripts is a directory containing a set of JDL files which could be submitted by `submit_jobs`.

The `submit_jobs` scripts run on `gridUI` and submit a set of JDL files from a location specified by the user to the GMS in a controlled manner. `submit_jobs` takes, as an argument, the user-input limit to how many grid jobs Grid Ireland permitted that user to have running simultaneously. It then submits jobs in batches of 10 at a time from `gridUI`, waiting to ensure that all jobs previously submitted have started running before submitting any more. It also monitors how many jobs are running at a time, stored as the variable `running_job_count`. If $(\text{running_job_count} + 10)$ is greater than the limit of simultaneous jobs, submission is halted until `running_job_count` is low enough that submitting a batch of new jobs would not cause it to exceed the limit. The output from this set of scripts is a text file containing a set of job identifiers which can be used by `check_jobs` to monitor the progress of those grid jobs.

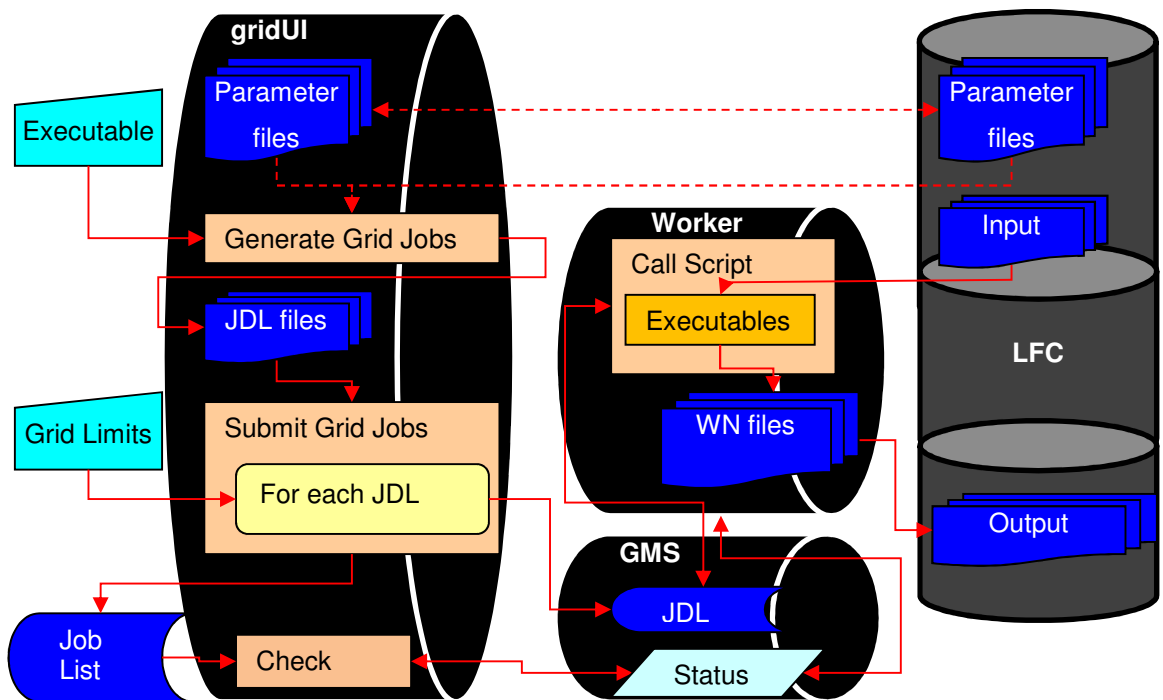


Figure 7-16: Grid Management Software

The Call scripts are unique to each grid job, and their details are discussed under the headings of those grid jobs above. In general, the Call scripts copy input from the LFC,

run executables which generate output on the WN, and then copy and register the output to the LFC in the appropriate locations.

Finally, `check_jobs` interfaces with the GMS and tracks the status of on-going grid jobs. The job lists output from `submit_jobs` is used as an input to `check_jobs`. It generates a list of job statuses which can be piped to a file or displayed on `std.out` as determined by the user. These statuses can also be piped through standard Unix tools such as `grep` and `wc` to isolate individual jobs or particular groups of job statuses.

7.2.5. SQL Queries to CAS

In this project, SQL queries to the SDSS Catalogue Archive Server (CAS) are used to produce CSV files which contain data used as an input to `parameterisation` in each of its three modes as discussed in Subsection 7.2.3. The CAS is a Microsoft SQL database which contains a variety of tables to store information about SDSS observations. [111] Queries to the CAS are used to generate CSV files containing the following information:

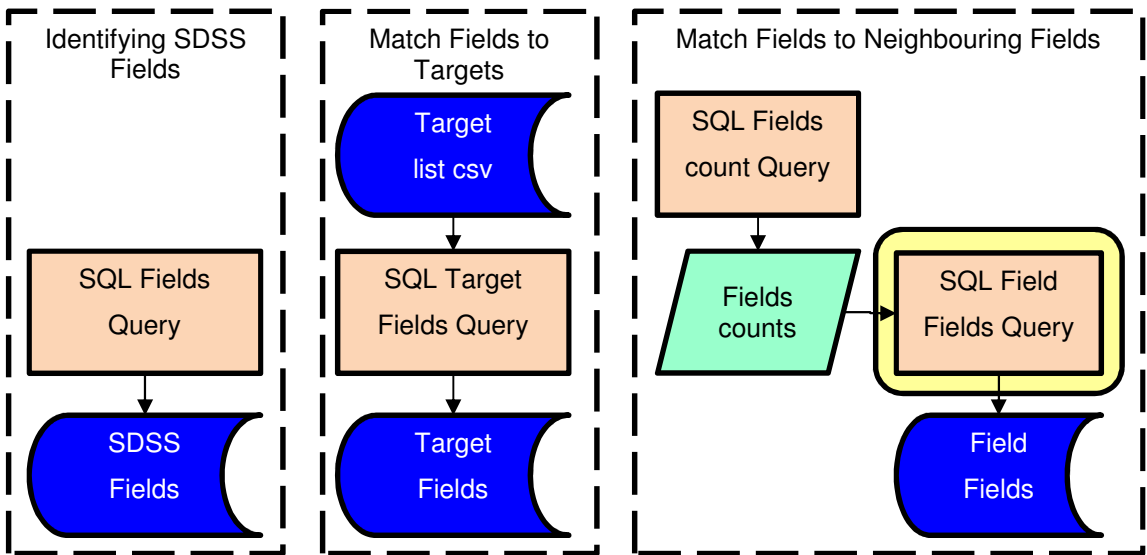


Figure 7-17: SQL Queries to generate input for Parameterisation

- A list of all fields in SDSS to be used with the API
- A list of all fields within the size of a FoV of each target in a target list to be used with the pipeline in Target List mode

- A list of all fields within the size of a FoV of the extreme points (maximum/minimum for each of RA and Dec) of each field in a catalogue for use with the pipeline in Catalogue Traversal mode

A suite of SQL queries has been developed by Dr. Eugene Hickey to generate these outputs. [159] The first query is to generate a list of fields in SDSS. This query selects all distinct combinations of the field identifiers `run`, `rerun`, `camcol` and `field` in the field table. Each such combination uniquely identifies a single SDSS field, and thus a single SDSS Calibrated Objects file as used in this project. The CSV files generated as output by this query are defined in Subsection 8.3.5.1

The second query requires a list of target objects. In the case of the quasar catalogue, this input list consisted of the list of objects in the SDSS quasar catalogue. [137] [161] With this input, for each target, the query selects all fields meeting the criteria shown in Equation 7-1

$$\begin{aligned}
 RA_{field,max} &> RA_{target} - \left(\frac{S_{FoV}}{\cos(Dec_{target})} \right) \\
 &\text{and} \\
 RA_{field,min} &< RA_{target} + \left(\frac{S_{FoV}}{\cos(Dec_{target})} \right) \\
 Dec_{field,max} &> Dec_{target} - S_{FoV} \quad \text{and} \quad Dec_{field,min} > Dec_{target} + S_{FoV}
 \end{aligned}$$

Equation 7-1: the fields which can be included in a FoV with a given target – those whose maximum coordinate is greater than the coordinate of the target less the size of the FoV (S_{FoV}) and whose minimum coordinate is less than the coordinate of the target plus the size of the FoV, with correction factors for RA as per Equation 5-1

The query then returns the target data, the field descriptors and a set of data used as control variables. The CSV files generate as output by this query are defined in Subsection 8.3.5.2

The final SQL Query acts as a nested SQL Query with two stages, as the scale of the query is beyond the capacity of the CAS to handle as a single query. Instead, a first query is run for each SDSS field, to count (but not identify) the number of fields in the catalogue and assign a sequential number to each field. The output of this query is used

as input to a sequence of iterations of a query that, for each target field, identifies the fields which met the criteria assigned in Equation 7-2:

$$\begin{aligned}
 RA_{field,max} &> RA_{TF,min} - \left(\frac{S_{FoV}}{\cos(Dec_{target})} \right) \\
 &\text{and} \\
 RA_{field,min} &< RA_{TF,max} + \left(\frac{S_{FoV}}{\cos(Dec_{TF,min})} \right) \\
 Dec_{field,max} &> Dec_{TF,min} - S_{FoV} \quad \text{and} \quad Dec_{field,min} > Dec_{TF,max} + S_{FoV}
 \end{aligned}$$

Equation 7-2: the fields which can be included in a FoV with a any point in a target field (TF) – those whose maximum coordinate is greater than the minimum coordinate of the TF less the size of the FoV (S_{FoV}) and whose minimum coordinate is less than the maximum coordinate of the TF plus the size of the FoV, with correction factors for RA as per Equation 5-1

This query, as defined in Subsection 8.3.5.3, returns the field descriptors for the Target Field and the fields used to create the mosaic around it, and a set of control variables.

7.2.6. Error Handling Routines

Error Checking, used throughout the C programs used in this project is implemented by means of a header file, which includes the error definitions given in Table 7-2 and a function contained in an independent c file which is *included* in the source code for each program.

Error Checking is carried out by capturing the return value of each function as it is called. When an operation which has the potential to produce an error is called, its return value is checked. When an exception occurs in a given operation, the function which called that operation returns an appropriate error as selected from the list in Table 7-2. Exceptions which occur using FITSIO commands return the FITSIO *status*, which works with the built-in FITSIO error checking routines.

This return value is passed to the `error_checking` function, which compares the error value against the possible error values. The first check is against `ERR_NO_ERROR`, defined to be 0, a value which is returned if the function completes successfully: this prevents unnecessary further checks.

The second check takes advantage of the fact that FITSIO standard errors are denoted by positive integers, while the errors defined in this project are denoted by negative errors. A positive error value is assessed using built-in FITSIO error checking routines.

Error Definition	Error Value	Error Description
Non- Errors (0)		
ERR_NO_ERROR	0	No Error present
API Errors -(1000 - 1999)		
ERR_OUT_OF_MEMORY	-1001	Memory full, unable to load more data
ERR_END_OF_FILE	-1002	Pointer reached end of file prematurely
ERR_FILE_READ	-1003	File Read Error occurred
ERR_CANNOT_OPEN	-1004	File Cannot be opened
ERR_CANNOT_CLOSE	-1005	File Cannot be closed
ERR_PAR_NOT_FOUND	-1006	Parameter not found in CTI
ERR_WRITE_FAILURE	-1007	Unable to write output file
ERR_INVALID_COLUMN	-1008	Invalid column name
ERR_COLUMNS_EQUAL	-1009	Two columns have the same name
ERR_FITS_ERROR	-1010	Non-specific FITSIO Library Error
Pipeline Errors -(2000 - 2999)		
ERR_NOT_FOUND	-2001	No viable output found for any target
Parameterisation Errors -(3000 - 3999)		
ERR_TOO_MANY_ARGS	-3000	Too many arguments submitted to parameterise
ERR_WRONG_FILE_TYPE	-3001	Input file is of the wrong type

Table 7-2: Error Definitions. Errors defined in this project use negative integers. FITSIO Built-in errors use positive integers to report errors

Following that, a set of nested *if/else* statements compare the error against the defined errors. If the error matches one of the known errors, `error_checking` prints an appropriate error message to `std.out`. If the error does not match any, an *else* statement prints “an unknown error occurred” and returns the error. If, after

error_checking is complete the error value is still not ERR_NO_ERROR, then the function returns the error.

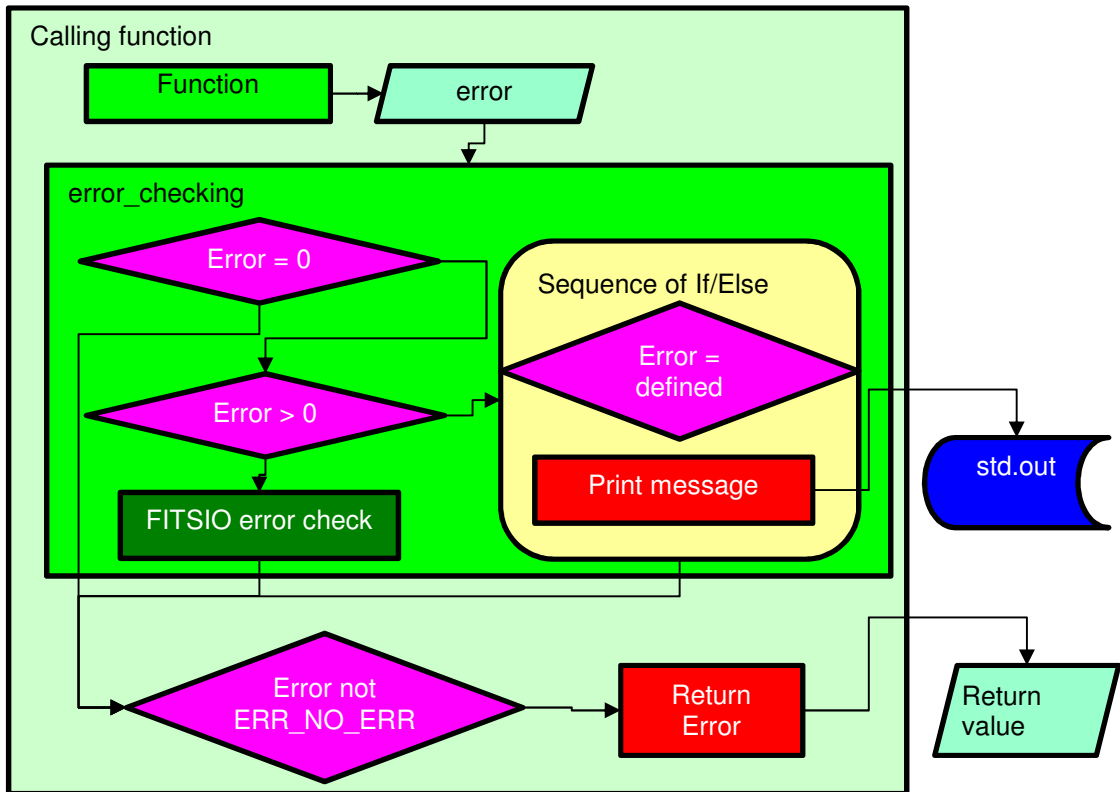


Figure 7-18: Error Checking System

7.3. Summary

The design of this project can be considered under two headings: constraints imposed upon it by external or prior concerns and a structured, Top-Down approach to describing the components of the system.

The constraints are based on the astronomical goals of the project and computational limits of the system. The astronomical goals of the system are to optimise pointings for differential photometry on given targets, analyse catalogues of potential targets, and generate output catalogues of these pointings. The solution to achieving these goals is to implement the Locus Algorithm using C programs with the FITSIO library to access and generate FITS files.

The computational limits of the system are derived from the final data format definitions, quantity of data and the available computing resources. The Data from

SDSS requires data cleansing to reduce data volume and isolate the required data for the project. This is achieved by use of the API to generate the Local Catalogue. The HPC solution used for this project is Grid Ireland, which operates under the `gLite` system, which requires that work be divided into grid jobs before it can be submitted to the grid using JDL files.

These constraints together impose a Bottom-Up flavour on the consequent design. From the Bottom-Up, the major components of the project are FITSIO routines run as part of C programs, bundled together with `gLite` commands in BASH shell scripts, which are run as part of grid jobs submitted by grid management scripts. The `gLite` commands interact with the LFC, where data and software needed for the project are stored. Data for the project is transferred into the LFC from the SDSS DAS by batch data transfer processes. Input from the user and from SQL scripts run on the SDSS CAS specify the behaviour of the grid management scripts.

From the Top-Down perspective, the project has two main components: the API and the Pipeline, and four components which serve the main ones: Parameterisation, Grid Management, SQL Queries to the CAS and Error Checking.

The API accesses SDSS data stored in the LFC, identifies the structure of that data with the Diagnose program and generates the reduced data set of the Local Catalogue in the Extract Program. This Local Catalogue is then stored in the LFC for use with the Pipeline.

The Pipeline takes input from the user regarding observational conditions and from CSV files generated by the CAS to identify the relevant Local Catalogue files. It then processes this data to generate the Output Catalogue. The Pipeline operates in two modes: Target List mode, in which the user supplies a list of targets, as was used to generate the Quasar Catalogue as shown in Subsection 11.3; and Catalogue Traversal mode, which was used to generate the Exoplanet Catalogue discussed in Subsection 11.4.

The Parameterisation software is, structurally, more like a set of modules used in both the API and Pipeline. It generates parameter files (PRM and PPR) which are used to manage job size in the two major components. In both cases, these parameter files list

the `.fit` files that are needed for that job. In the case of the PPR (Pipeline Parameter) files, the file also includes a list of the targets that are to be processed by the Pipeline job. These are generated based on user input and CSV files output by the SDSS CAS based on input in the form of SQL queries. These SQL queries come in three different varieties, one for each of the three modes of operation of the parameterisation software.

The Grid Management Software is a suite of tools which are used, with subtle variations, to Generate, Submit, Call and Monitor grid jobs in the other three components of the project. Each of those four script families is used as a module in the grid enabling of the other project components.

Throughout the C programs used in the project, a suite of Error Checking software is used to determine if and when any exceptions occur in the programs, and if they do, to report on those errors.

As Jalote (2005) [144] suggests is true of most real software projects, this project incorporates elements of both Top-Down and Bottom-Up design. The blending of these two approaches allows for the incorporation of Bottom-Up requirements and constraints, while the Top-Down structural approach creates a coherent design into which modules can be placed over time as the project evolves.

8. Data Storage and Management

This Chapter discusses the data storage systems and structures under three headings

- **Storage Devices**

Section 8.1 discusses the nature of the various computing and data storage devices used in this project, and how those elements interact with one another and the user. These elements consist of the Development Environment, gridUI, the LFC, the GMS and the individual WNs.

- **Directory Structure**

The data is stored in a hierarchical structure of directories as described in Section 8.2. The same structure is used on every storage system within the project. This structure is used to ensure that data can be reliably accessed on any computing element of the project.

- **Structure of files**

The internal structure of each of the file types used in the project is fully described in Section 8.3. Some of the file types used are defined externally (e.g. CSV, FITS) and some are novel to this project, and are designed internally. (e.g. PRM, PPR)

8.1. Data Storage Devices

There are five primary data storage environments used in this project, each with a distinct function and set of interactions with the other elements. These devices are listed below. The interactions between these elements are outlined in Figure 8-1 and detailed below it.

- **Development Environment (DE)**, which provides a structured environment within which to develop software, particularly the C programs used in this project
- **gridUI**, a dedicated computer located at ITTD which acts as a gateway through which users could interact with the grid to upload data, run and monitor jobs and retrieve output
- **Grid Management System (GMS)**, a system which processes grid commands and which allows for data and grid jobs to be transferred between grid elements

- Logical File Catalogue (LFC), which is a distributed storage system which permits long-term storage of large amounts of data which is discussed in Subsection 8.1.1
- Worker Nodes (WNs), which process grid jobs.

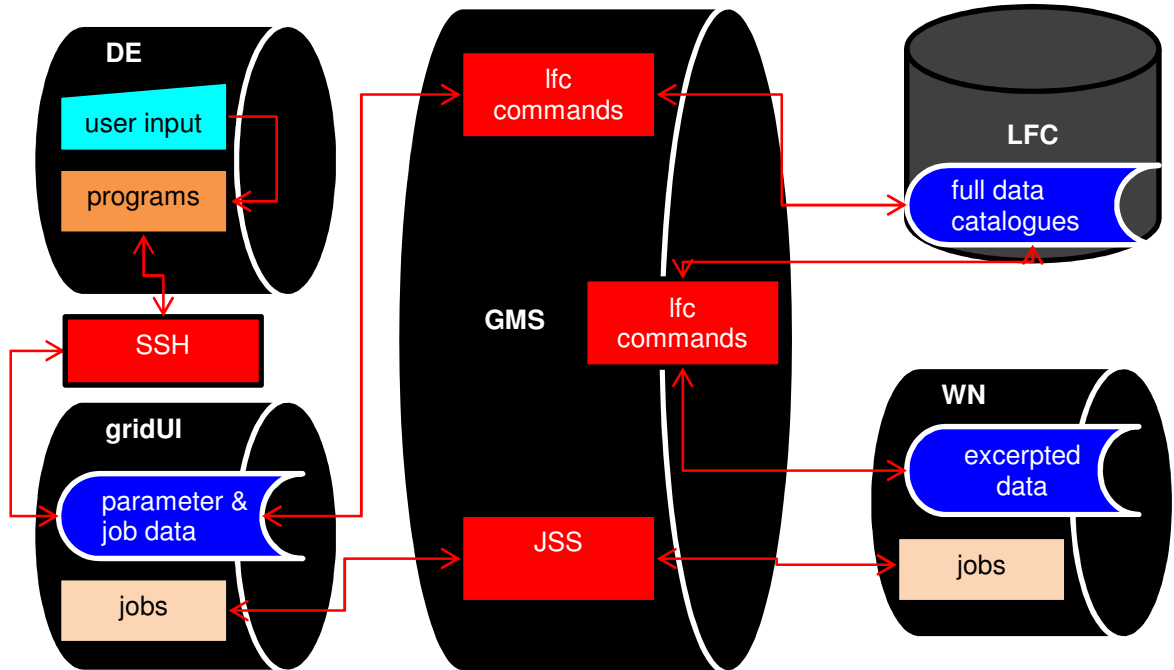


Figure 8-1: The data storage and processing elements used within the project, and the interactions between them.

The Development Environment (DE) is a VirtualBox Virtual Machine (VM) operating Scientific Linux version 4.5 and running the Eclipse Europa Integrated Development Environment (IDE.) Virtualisation of the DE permits it to be ported between physical machines as needed. Programs are written and tested in this environment without grid interactivity, and are not dependent on external resources. The interface between the Development Environment and gridUI is provided by Secure Shell (SSH), a network protocol which provides for secure transfer of both user input data and executable programs. [162] The DE does not have direct access to any other storage elements.

The Grid Ireland User Interface (gridUI) is the means by which the user interacts with the grid. It is usually operated through SSH from the DE, although, for convenience throughout this project, user input transferred from the DE through SSH to is described as user input to gridUI, unless explicitly specified otherwise, as this simplifies explanations.

A communications link exists between gridUI and the GMS. Communication between these two systems takes the form of one of the following two categories of command:

- Logical File Catalogue (LFC) commands mimic standard UNIX commands [7] as discussed in 3.2.1, and are primarily used to copy data to and from the LFC.
- Job Submission System (JSS) commands are used to submit, monitor and retrieve grid jobs.

WNs are the individual machines to which the GMS assigns grid jobs. WNs which are part of Grid Ireland are hosted by a number of institutions around the country. Each has a UNIX operating system and local storage space. Specific features of the WN are specified as a requirement of a given grid job within the text of the JDL file submitted to the GMS.

8.1.1. Logical File Catalogue

The purpose of the LFC in this project is to serve as a long-term archive. As discussed in Subsection 3.2.1, the LFC is a distributed storage system used by a number of projects in collaboration with Grid Ireland with 700 Terabytes of storage space. [93] It is not necessary for the user to know the physical location of the storage elements used for a piece of data. Rather, the GMS allows files to be managed and accessed by means of their Logical File Name (LFN.) [7] LFNs must obey the naming conventions of the UNIX directory and file name system.

Data on the LFC cannot be directly accessed by programs, scripts or the user. Instead, requests are sent to the GMS, through a set of commands known as `gLite` commands, which are incorporated into scripts for automated data retrieval and storage from either gridUI or a WN as discussed in Subsection 7.2.4.

Finally, standard operating procedure for Grid Ireland is that users are granted permissions for access only to specific Sections of the LFC by virtue of their membership of a given Virtual Organization (VO.) Designated areas for a particular VO are identified with parts of the LFC/LFN directory structure. The SCG are members of Cosmo VO, and therefore `/grid/cosmo/` is the working directory used.

8.1.2. Grid Data Transfers

Jobs submitted to the GMS are apportioned to WNs. Further JSS commands are used to monitor the status of grid jobs and are used to retrieve the `std.out` and `std.err` output from grid jobs, which are stored as files on the GMS. Files produced as output from grid jobs are normally copied to the LFC, and are not accessible directly in this way. [7]

Each grid job is a “clean” start – the WNs do not retain data or programs from one job to the next. Data and programs therefore have to be copied from a remote location for each grid job. In this project, data is stored on the LFC and copied to the nodes using LFC commands incorporated into the scripts as and when the data is needed.

The `std.out` and `std.err` output from grid jobs are returned to the JSS as part of the grid job process. [7] Output data files are copied to the LFC using LFC commands for later retrieval.

The GMS acts as an intermediary between the LFC, gridUI and the WNs. The user does not have direct control over or access to the GMS, but rather they submit requests to it in accordance with the gLite users guide. [7] As with SSH access from the DE to gridUI, the intermediary role of the GMS is usually not discussed in terms of LFC or JSS commands. Instead, these commands are treated as direct interactions between the three elements except where relevant.

As can be seen from this, data is regularly passed from one storage element to another. Each of these elements has its own directory structure. This structure is not known to the user at run-time and has the potential to be different from one WN to another. Therefore it is necessary to develop a solution for data access that works on many nodes.

The solution, as described in Section 8.2, is to use a directory structure built on a relative path from the working directory of the computing element. From that working directory, a new directory tree is created in every instance to hold the data needed on that occasion. By working with a relative path from the working directory, it is possible to use the same file paths at every stage of the project without needing to know the

working directory. In addition, it makes unit testing easier as a completely new environment can be simulated simply by creating a test directory and changing working directory to that.

8.2. Directory Structure

A rigid directory structure allows any instance of a particular piece of data or program to be stored in the same place in the structure regardless as to whether it is stored on gridUI, a WN or in the LFC. Note that the entire data structure is not recreated in each instance, just a subset of it as indicated Figure 8-1: those paths that are needed for that job.

When a piece of data is copied to a storage element, the directory in which the data is to be stored in has to be created first. The Unix command `mkdir -p` and its gLite analogue `lfc-mkdir -p` are used to create the directory on that element before the data is copied. [7] If the directory has already been created, these commands have no effect, but they are required to prevent file copying failures.

8.2.1. Top Level Structure

At the top of the data structure is the physical storage. The exact nature of this physical storage varies from element to element as outlined in Section 8.1. Each computing element opens to a default working directory (e.g. `/home/creanero/` on gridUI.) By design, the user does not need to know this directory. Instead, the remainder of the directory tree is constructed from there in each instance.

From the working directory, the first two layers of the directory structure are designed for extensibility to future projects. The first is a directory for all ITTD SCG projects called `ittd`. Beneath that, there is a subdirectory, namely `grid_cdm`, for this project – Grid-based Catalogue Data Mining. Future projects established within the SCG may expand upon this structure.

Within this project, there are five primary directories, `workspace`, `scripts`, `test`, `data` and `release`. Those directories are each expanded upon in their own Subsection below.

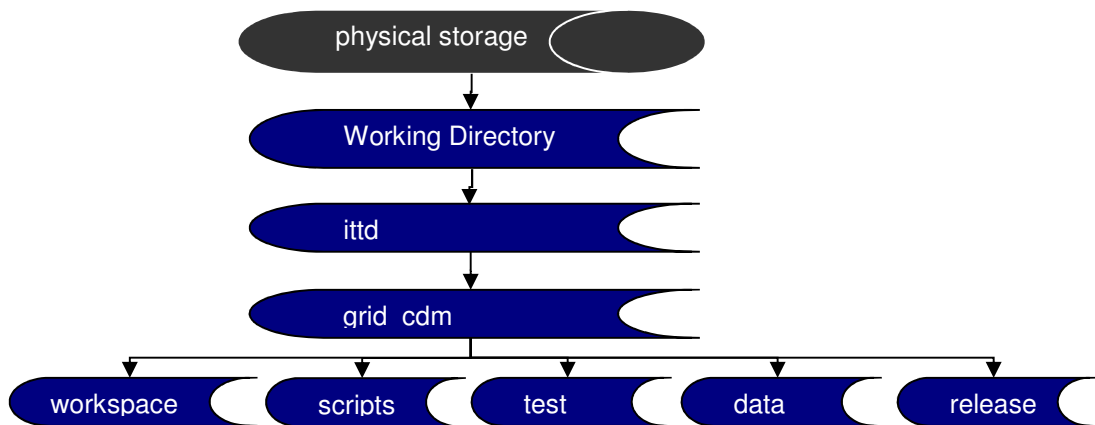


Figure 8-2 Top Level Directory Structure: This structure is used on all physical storage elements, and constructed as needed below the working directory on that element.

8.2.2. Workspace

The `workspace` directory is where C code is developed, compiled, debugged and stored. This directory is used primarily within the DE. Within this directory, individual workspaces exist for each top level version of the code that was developed. When a major new code revision is started, a new workspace is created and the files from the previous edition are copied to that directory before being modified. This version control allows for reversion to an older version of the software if needed as discussed in Subsection 6.2.2.2.

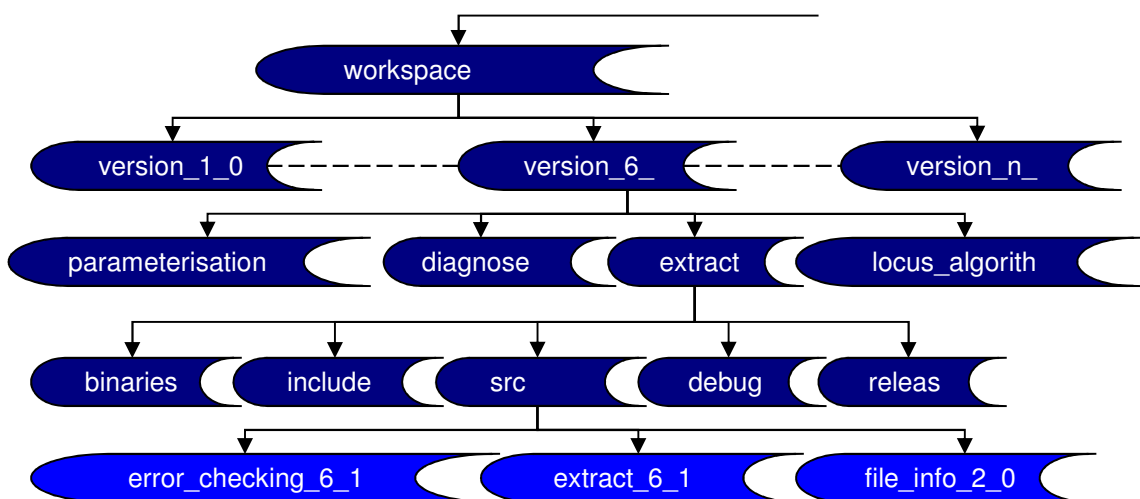


Figure 8-3 Workspace Directory: The top level of subdirectories is used for overall version control. Lower levels are automatically generated by the Eclipse IDE.

Within these workspaces, the directory structure is the default structure created by the Eclipse IDE. In each version, a directory is created for each of the C programs used in the project. As can be seen from Figure 8-3, the final version (version 6.1) has four main programs, and thus four subdirectories: `parameterisation`, `diagnose`, `extract` and `locus_algorithm`.

As developed by Eclipse, each program has several directories of components used to build the final executable as illustrated with the example of the `extract` program in Figure 8-3. The `binaries` and `includes` folders are automatically generated by Eclipse and are used when compiling the code. The `src` directory contains the source code for the program. The `debug` folder contains a version of the compiled program optimised for debugging, together with other debugging information such as breakpoints. The `release` directory is where the final version of the program is written to, after debugging is complete. It is this executable that is copied to the main `release` directory (see 8.2.6) once the program is fully tested and ready for use.

8.2.3. Scripts

The Scripts folder is used to contain the scripts used for grid management as discussed in Subsection 7.2.4. These scripts are organised into subdirectories for the various functions they performed. For example, the directory shown in the diagram (`csv_management`) is used to contain scripts used to manipulate and modify the CSV files used in the project.

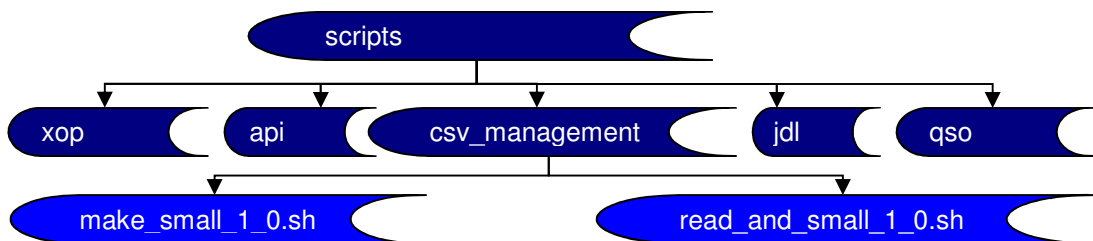


Figure 8-4: Scripts Directory: Subdirectories are used for families of scripts.

As discussed in Subsection 6.2.2.2, version control takes the form of modifying the file names with the version numbers as and when a new version of the script is implemented.

8.2.4. Test

The test folder is used to test scripts, programs, data and other components of the project on gridUI. This allows for tests to be carried out in an environment with access to the GMS. The first sub-level of this directory is designed to be changed on a daily basis to hold batches of tests carried out on a given date. Each individual test is given its own “run” directory below that level and necessary files are copied into it depending on the test. The test is then run by the user, and depending on the nature of the test, it creates a partial replica of the overall directory structure below that run directory as shown in Figure 8-5.

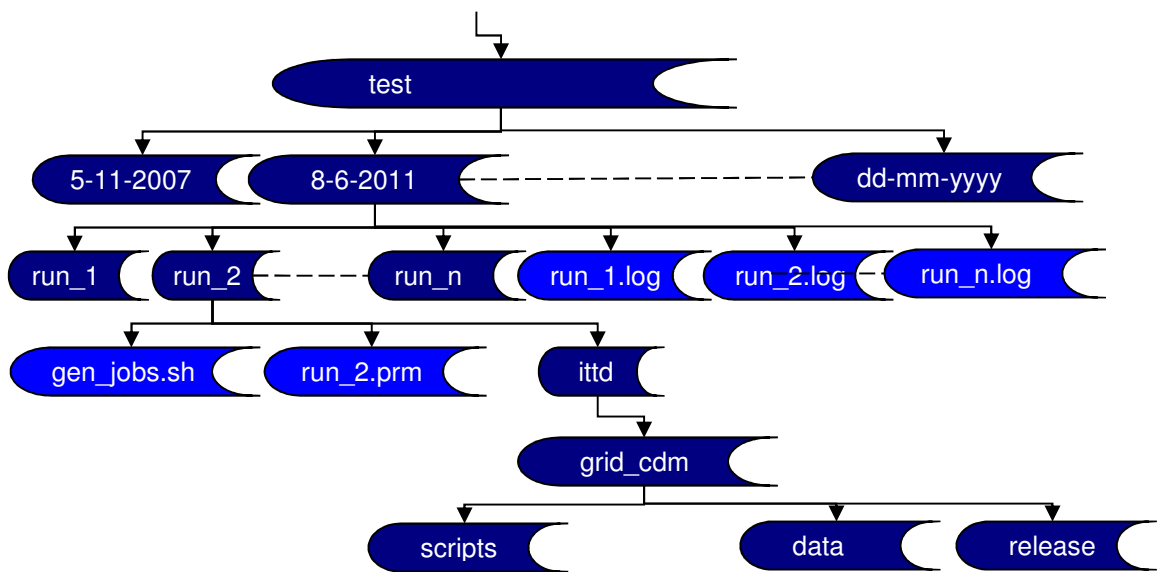


Figure 8-5: Test Directory: a partial directory tree for the test directory. A sample test (test 2 of 8-6-2011) is shown with an expanded directory structure. As can be seen from this example, a given test does not necessarily generate a full directory structure

Only the required components of the directory structure are copied in this situation, as was the case during a grid job. Typically, for example, a test job does not require a test folder underneath it.

In most cases a log file is created. For ease of access, the convention is that all logs for a single day were redirected into that day’s directory, not to the individual test’s directory, so that there is no risk of the log file being compromised by the run, or the run by the log file.

8.2.5. Data

The `data` directory is the largest directory in the overall structure. This directory contains the catalogues, parameter files and output files that are used and generated in the project, and each of these has its own corresponding subdirectory.

The `catalogues` subdirectory is designed for extensibility to other catalogues as proposed in Subsection 14.1.5, however SDSS is the only catalogue in *this* project. The original SDSS files are stored in a directory called `raw`, while the local catalogue counterparts to each are stored in a directory called `local`. The unique SDSS directory structure was retained for this project and is described in Subsection 8.2.5.1. The directory structure is a deep, Top-Down structure that is defined by SDSS to allow files to be found by using the CAS.

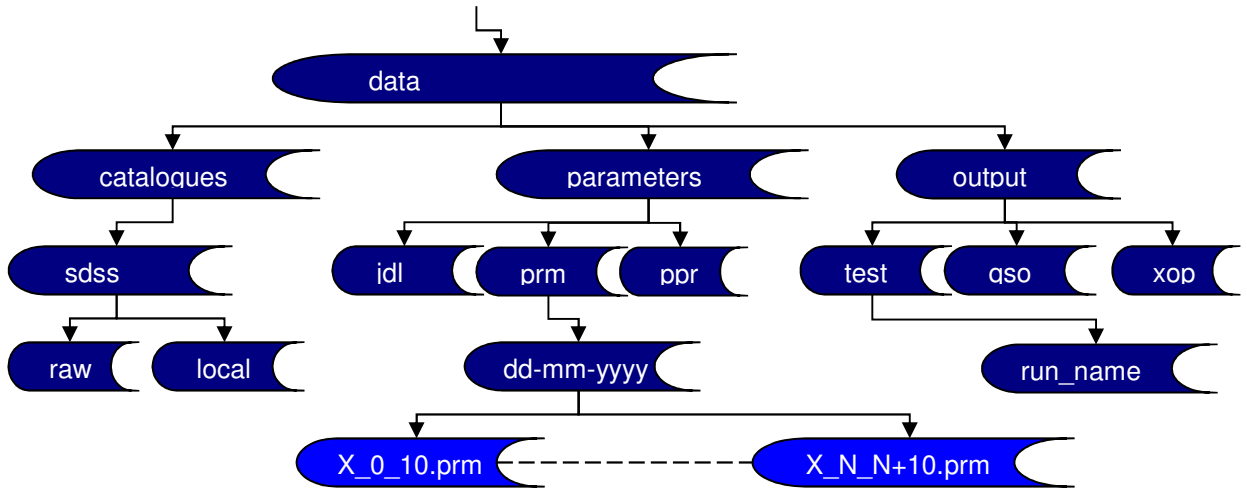


Figure 8-6: Data Folder: The SDSS Catalogue structure is shown in Figure 8-7. The subdirectory structure of the parameter directories and the naming convention of the files are the same in the `jdl`, `prm` and `ppr` directories. Output catalogues in `test`, `xop` and `qso` directories are typically stored in a directory with their date, and may include further information in the directory or file name as shown in Table 8-1.

The parameter files are generated automatically by the parameterisation program (see Subsection 7.2.3) and stored in the `parameters` directory. Within that directory are directories for PRM files (used in the API stage of the software, defined in Subsection 8.3.3.1), PRPR files (used in the pipeline phase of the software, see 8.3.3.2) and JDL files (which are used to submit jobs to the grid see 8.3.6).

Within the `ppr` and `prm` directories, separate directories are created by the user for each set of parameters used for a given batch of jobs. These are usually identified by date, and may include other information as needed, encoded in the file name (e.g. the filenames for test parameter files were preceded with `test_`). As parameter files are generated automatically, they are given sequential names to reflect the targets or fields that their contents refer to.

FoV	Resolution	Max Magnitude Difference	Max Colour Difference	Colour band
0.25 degrees	1 arcsecond = 0.000277 degrees	2 mag	0.1 mag	r
File name	xop-0_25-0_000277-2_0-0_1-r-0-100_output.fit			

Table 8-1: Illustration of the encoding of exoplanet job parameters in an output file name. Colours are used to indicate the data components to which each component of the filename corresponds

JDL files are stored in a directory structure that mimics that of the PRM or PPR files that they are generated to submit, but with the “`ppr`” or “`prm`” level of the directory path replaced with “`jdl`” as shown in Figure 8-6. The name of a JDL file is the name of the PRM or PPR file it corresponds to with the extension changed.

Output files are the results of the project. They are stored in the `output` folder, in an appropriate subdirectory for the targets that they reflect. The two primary jobs are the Quasar and Exoplanet jobs, labelled with `qso` and `xop` respectively. The outputs from test jobs are stored in the `test` subdirectory. The names of the output files from a particular job are typically encoded with the arguments used for that job as illustrated in Table 8-1.

8.2.5.1 SDSS data structure

Within the data folder, the entire SDSS DR 5 object catalogue is stored in a directory tree. This tree mimicks the SDSS data storage model, which allows for access to the data by reference to the results of SQL queries submitted to the CAS as shown in Subsection 7.2.5.

The SDSS model incorporates four pieces of critical information about a file into the file name and the directory structure as shown below. [158] This model is retained in this project, and used to store both the raw SDSS Catalogue and the Local Catalogue.

Excerpted from the SDSS Glossary (SDSS, 2007)

“Run: A Run is a length of a strip observed in a single continuous observing scan. A strip covers a great circle region from pole to pole; this cannot be observed in one pass. The fraction of a strip observed at one time (limited by observing conditions) is a Run. Runs can (and usually do) overlap at the ends.

Rerun: A reprocessing of an imaging run. The underlying imaging data are the same, just the software version and calibration may have changed.

Camcol: A Camcol is the output of one camera column of CCD's (each with a different filter) as part of a Run. Therefore, 1 Camcol = 1/6 of a Run.

Field: A field is a part of a camcol that is processed by the Photo pipeline at one time. A field consists of the frames in the 5 filters for the same part of the sky. Fields overlap each other.” [158]

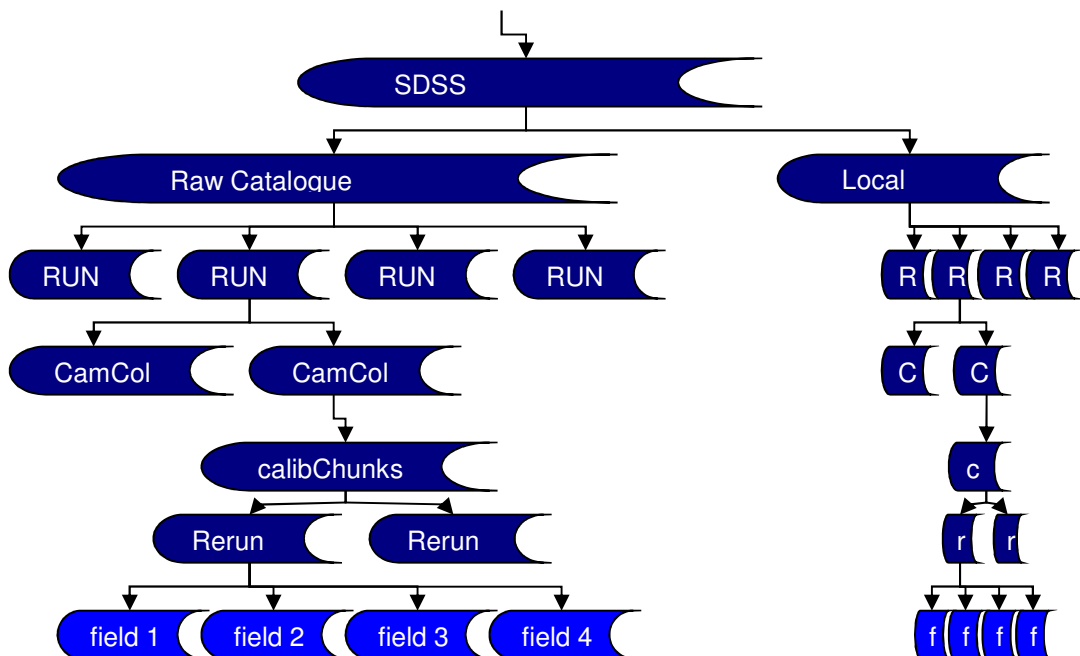


Figure 8-7: SDSS Directory structure: As is shown in this diagram, the same directory structure is used for both Local Catalogue and Raw SDSS catalogue files.

From these pieces of information it is possible to locate the position on the sky of any given file, and conversely for a given position, to identify the file appropriate to that position using the CAS. Each of the four pieces of information above takes the form of an integer which is used to create the file paths as shown in Figure 8-8.

R/r/calibChunks/C/tsObj-RRRRRR-C-r-FFFF.fit
1458/40/calibChunks/4/tsObj-001458-4-40-0352.fit

Figure 8-8: Definition of an SDSS tsObj File Path. Given **Run (R)**, **Rerun (r)**, **Camcol (C)** and **Field (F)**: Rerun and Field in the filename are padded with leading zeroes to the length illustrated, but leading zeroes are not used in the directory names, nor are they used for Rerun or Camcol in either the filename or the directory. [121] An arbitrarily selected example is shown.

8.2.6. Release

A version of the `release` directory is maintained on the LFC and contains the release versions of the novel executables generated during the project. Each version of the software is stored in a separate directory, and each program is stored in its own subdirectory. When an executable file is required for a job, it is copied from the LFC to a duplicate `release` directory on the computing element it is to be used on, and executed from there using its relative path.

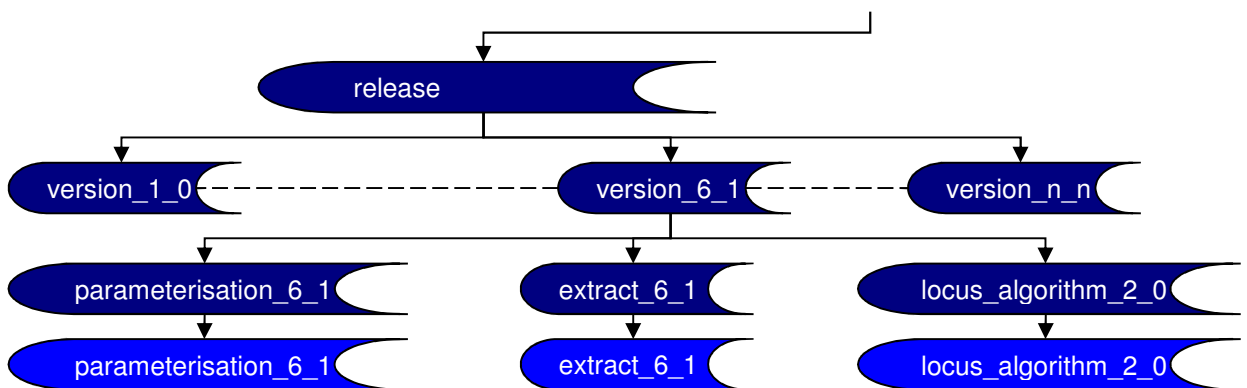


Figure 8-9 Release Folder

8.3. Internal Structure of File Types

Throughout the project, a number of file types are used. For some of those file types, the data models are original to this project, and others are externally defined. Knowing the data model of each file type is essential to the development of code to access and manipulate those files.

In this Section, each file type used in the project is described in both function and composition. In some cases, the data structure is flexible, and may have permitted data to be stored in several ways, for example, by use of data headers. In these cases, the general model is described first, and then the specific structures used in this project are individually described. The files used in this project can be categorised into two broad categories: Binary data files, and text files.

The main types of binary data file used in this project were

- FITS files, (extension `.fit` or `.fits`) an international standard in the astronomical community for transfer of data and images. [126]
- CTI files, (extension `.cti`) an internally defined file type used to describe the contents of FITS files as part of the API
- Parameter files, a set of two internally defined file types: PRM and PPR (extensions `.prm` and `.ppr`) used to control the operation of the software developed for this project.

Text files are usually stored in ASCII encoding for portability. The text files used in this project are categorised as

- Unstructured Text files, used to contain text for input to or output from components of the project. Source code, shell scripts etc. are also encoded as text files.
- CSV files, (extension `.csv`) a *de facto* standard [163] used to store data in a table-like structure, where rows of the table are delineated by newline characters, and where the columns of the table are defined in a header (the first row of the file), and delimited by a distinctive character, usually a comma.

- JDL files, (extension `.jdl`) the standard format used to describe a grid job in the `gLite` system [7], where requirements, executables and arguments for the grid job are specified as a series of name-value pairs.

8.3.1. Flexible Image Transport System (FITS)

FITS stands for Flexible Image Transport System. FITS files are an international standard in the astronomical community. [164] FITS files can be used for data tables, images, data cubes or even higher-order data structures. FITS files are composed of sequences of HDUs (Header Data Units.) These HDUs are split into headers followed by data. FITS files are used for data transfer between organisations whose internal formats, hardware etc. vary. [86]

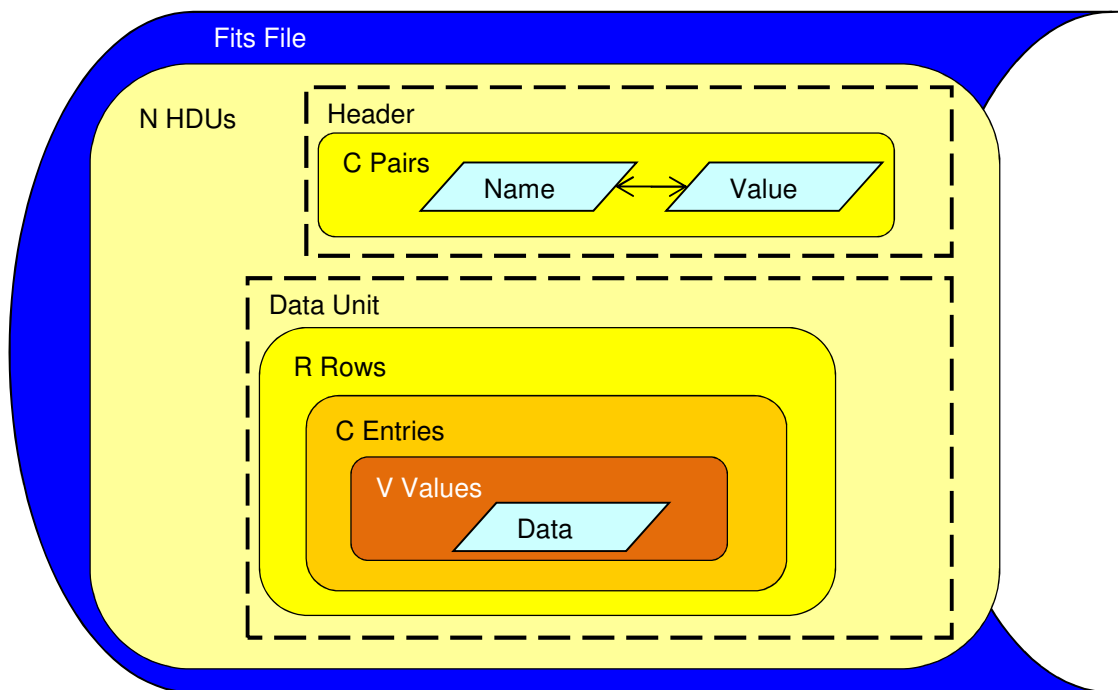


Figure 8-10: General Structure of a FITS File. Each fits file consists of one or more HDU. The HDU can be separated in to the Header and the Data Unit. The Header may contain zero or or more name-value pairs which describe the data in the Data unit. The data unit consists of zero or more rows each consisting of a series of column entries corresponding to the name-value pairs in the Header. The data in these columns may be a single entry or a vector of multiple values.

Headers consist of keyword=value statements. Headers describe the organisation of the data and the format of the contents. Headers may also include additional information about the instruments used or the history of the data. [126]

Data follows on from the header, and is structured as it is described in the header. Although the “I” in FITS stands for Image, FITS files have been adapted to transfer other types of data. [126] The first HDU in a FITS file must contain an array of one or more dimensions, but subsequent HDUs, called extensions, may be structured differently, within certain limits. A HDU need not contain any data at all. [126] The Data in a FITS image or data array must be in one of five data types, shown in Table 8-2.

Note that in this Subsection, when defining the structure of a specific FITS file or group of files, such as the SDSS object catalogue files discussed in Subsection 8.3.1.1, the contents of the header are not usually explicitly described. Instead, the contents of the header are considered to be implied by the description of the contents and structure of the data unit.

As discussed in Subsection 3.3.2.2, CFITSIO is a library of file access and manipulation routines for C which is designed to allow C programs to access FITS files. [164] It is maintained by NASA to be up to date with current needs of the astronomical community. CFITSIO is used extensively in this project. Similar libraries are available for FORTRAN and interfaces exist for other languages as discussed in Subsection 3.3.2.2. [126]

Description	Short name	Detail
Unsigned integer	unsigned int	8-bit unsigned binary integers
Signed integer	int	16-bit two's-complement signed binary integers
Large signed integer	long	32-bit two's-complement signed binary integers
Floating point number	float	32-bit IEEE-754 standard floating point numbers
Large floating point number	double	64-bit IEEE-754 floating point numbers

Table 8-2: FITS Data Types [126]

8.3.1.1 SDSS - `tsObj`

The FITS files supplied by SDSS for use in this project are the Object Catalogue files, also known as `tsObj` files after the first five characters of the file name for each. [121] Each object detected in an image taken by SDSS is recorded as a row in a table in the Catalogue. The `tsObj` files had a measured mean size of 11.8MB, and the full SDSS DR5 object catalogue was approximately 4.76 TB in size as stored in the LFC.

Object Catalogue files contain an entry for each observation recorded by SDSS, rather than a unique entry for each astronomical object. In addition, SDSS fields overlap, due in part to the shape of the sky, and for observational reasons. [43] As such, some objects are detected in more than one field, and thus are present in more than one catalogue entry. Finally, SDSS includes detections of objects which are not stars, including galaxies and other extended objects, and also including false detections and transient phenomena. [157]

For each object, there are 146 columns, which describe that observation of that object. [121] In each of these columns is a varying amount of data, ranging from individual long integers (e.g. to identify the field by `run`, `rerun`, `camcol` and `field` as discussed in Subsection 8.2.5.1) to arrays of five double-precision floating point numbers (e.g. to identify the five model magnitude values in each of the 5 SDSS colour filters `u`, `g`, `r`, `i` and `z`.)

Integers (specifically `status`, `flags` and `flags2`) are used by SDSS as 32-bit bitmasks of binary flags, which are used to describe each entry: for example, identifying if an entry refers to the primary observation of a particular object or not. [157] These flags were used by the Extract program as shown in Subsection 7.2.1.2 to identify the entries that corresponded to what SDSS, 2006, defined as a clean sample of stars. [157]

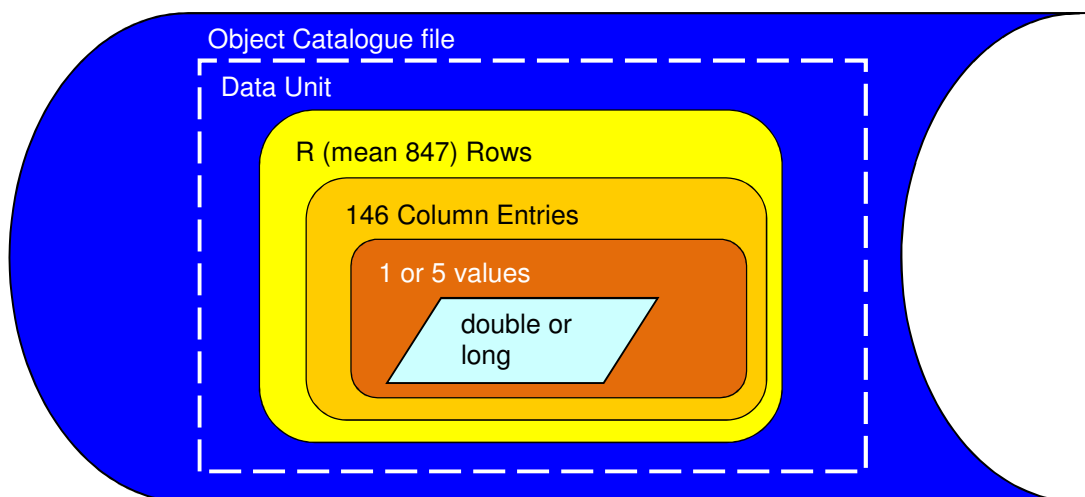


Figure 8-11 SDSS `tsObj*.fit` File: Only one HDU is used, which contains a mean of 847 rows and 146 columns of data. These columns contain entries which have either one value (referring to the observation as a whole) or five values, one for each SDSS band filter

8.3.1.2 Local

In addition to being filtered to include only the clean sample of stars, only a particular subset of information about each star is needed for the Locus Algorithm. Thus, Right Ascension (RA), Declination (Dec) and Magnitude (Mag) are recorded in the Local Catalogue, while the other values are discarded as part of the Extract process discussed in Subsection 7.2.1.2.

RA and Dec have unique values per observation, but there are several options for Mag in the SDSS Catalogue, each calculated using slightly different methods. SDSS provides a “model” value for magnitude, which is defined by SDSS to be the better of the De Vaucouleurs and exponential methods. [121] It is this value that is used throughout this project, and except where the distinction is relevant, it is referred to as “magnitude” or “mag” without qualification throughout this Thesis.

The reduction in both rows and columns provided for a substantial reduction in data volume. While the mean number of rows in each SDSS fit file is 847, the mean number of rows in a local catalogue file is 240. On occasion there are no entries in a local catalogue file as the entire field is considered a “secondary” image. This typically occurs towards the end of a stripe as fields began to overlap. [5]

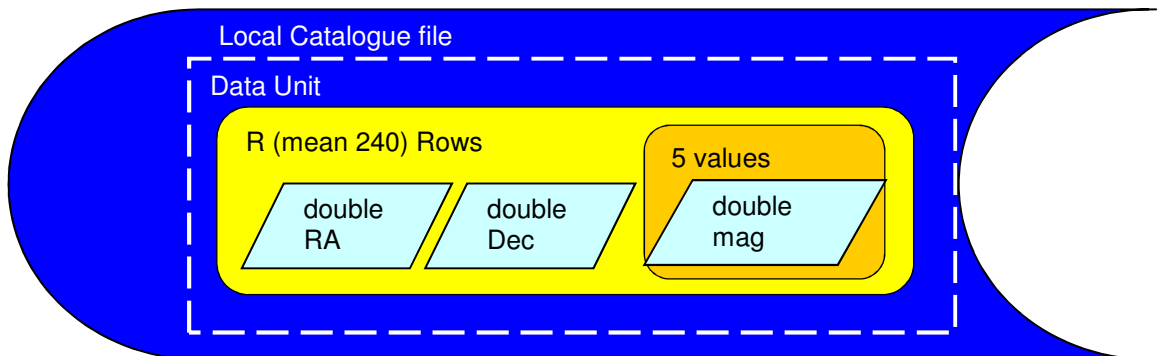


Figure 8-12: Local Catalogue FITS File. The local catalogue files contain RA, Dec and a vector of 5 magnitude values for each star that passes the extract process shown in Subsection 7.2.1.2

As a result of this reduction, Local FITS files are measured to be 584 times smaller (20.4kB as opposed to 11.8MB) on average than the original SDSS files they were derived from, and the entire Local Catalogue was measured to be 6.89GB, as opposed to

4.76TB for the SDSS catalogue. These catalogue results are discussed further in Chapter 11.

8.3.1.3 Output

The output files are used to store the output Quasar and Exoplanet Catalogues, the primary products of this project, as discussed in detail in Chapters 11 and 12. Output files are stored in FITS format for accessibility and portability, in keeping with standard practice in Astronomy.

In the output files, the data for the chosen target or targets is presented in the same manner as that in the Local fits files i.e. RA, Dec and Magnitude (as before, a vector of 5 values). The result for that target is presented in the form of the RA and Dec of the pointing and the score for that pointing. These six values constitute a row of the output catalogue.

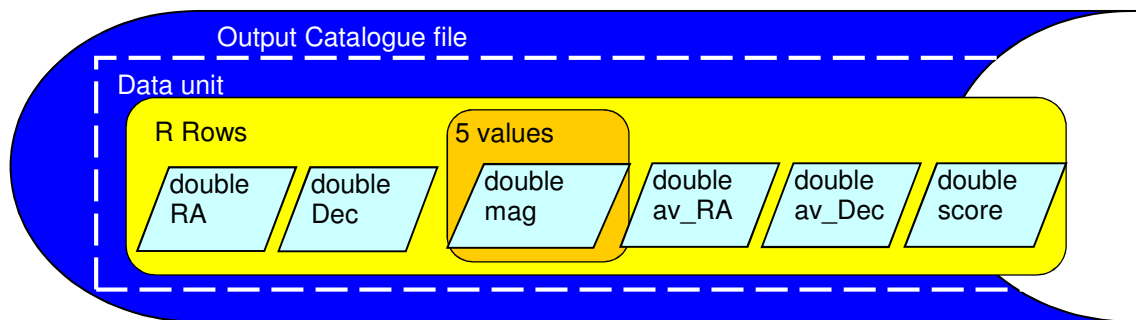


Figure 8-13: Output FITS File: Output data included RA, Dec and an array of mag as per the Local Catalogue. In addition, the RA and Dec of the pointing (av_ra and av_dec), and the score for that pointing are given.

Each file in the Quasar Catalogue holds 1,000 such entries, and while entries in the Exoplanet Catalogue vary between 0 and 175,688 there are an average of 41,955 entries per Exoplanet output file. Each file in the Quasar Catalogue is 87.8 kB in size, while the mean file size for the Exoplanet Catalogue is 3.22 MB. The Quasar Catalogue consists of 40 such files, while the Exoplanet catalogue is composed of 1,598 files. The total size of the Quasar Catalogue is 3.43 MB while the Exoplanet Catalogue totals 5.02 GB. More detailed data metrics on the output catalogues are provided in Subsection 13.1.3.

8.3.2. Catalogue Information (CTI)

CTI files describe the contents of FITS table files in such a way as to make their contents readily accessible to the Extract program. They describe the FITS table in terms of the number of rows and columns in it, and then list the column number, a 16-character column name, and three pieces of information used in the FITSIO interface: `typecode`, `repeat` and `width`.

CTI files are generated by the Diagnose program (7.2.1.1) and are used by the Extract program (7.2.1.2) to find the relevant data from a submitted FITS file. CTI files form a layer of abstraction, whereby the Extract program does not need to have the correct column for the required variables hard-coded, but rather can search for them in the CTI files, which may be useful for extensibility.

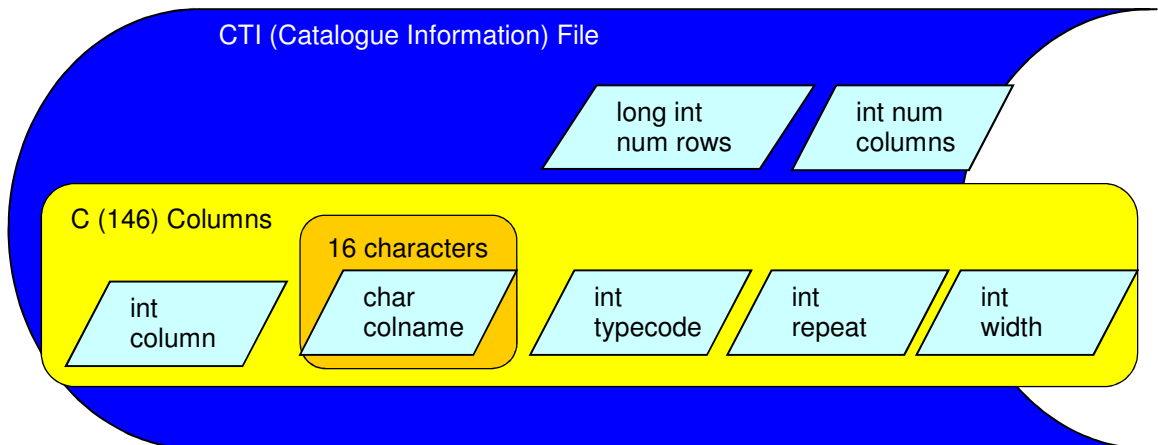


Figure 8-14: CTI File: each CTI file describes the contents of a FITS file. It lists the number of rows in the file as a long int, the number of columns as an int. It then has a repeating structure for each of the 146 columns in an SDSS fits file consisting of an int for the column number, and an array of 16 characters for the column name, and three more integers for the typecode, repeat and width of that column.

8.3.3. Parameter Files

Parameter files are generated by the parameterise program as described in Subsection 7.2.3. Parameter files are used to describe jobs to the executable files in the pipeline and API stages of the program. There are two types of parameter file, API parameter files, given the file type `.prm`, and pipeline parameter files, with the extension `.ppr`.

8.3.3.1 API Parameters (PRM)

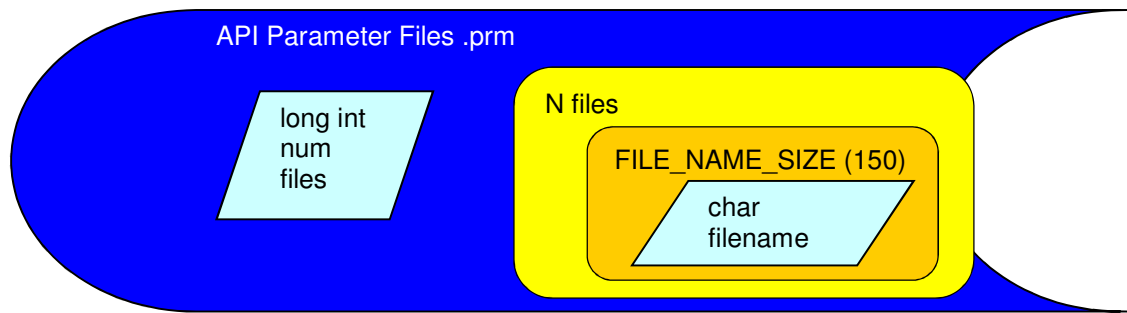


Figure 8-15 API Parameter Files (PRM): API parameter files consist of a long int representing the number of files to be processed as part of the job the file represents, and an array whose entries are arrays of characters – the path to each file. The length of these paths is a variable, defined to be 150

API Parameter files are used in two circumstances:

- To define batches of SDSS `tsObj` files used in the API to generate the Local Catalogue as described in Subsection 7.2.3.1
- As part of the parameterisation stage to create PPR jobs by batch as described in Subsection 7.2.3.3.

They control the job by listing the number of files to be processed in a `long int` control variable, and a relative path to each file. The subsequent programs can thus access the files using those paths, and iterate a number of times equal to the control variable.

8.3.3.2 Pipeline Parameters (PPR)

Pipeline parameter files are used to define pipeline jobs used to generate output catalogue files. They are used to convey the following information

- User parameters relating to the observation site the catalogue was optimised for
- Paths to the files needed to generate mosaics
- Information about the targets contained within those mosaics

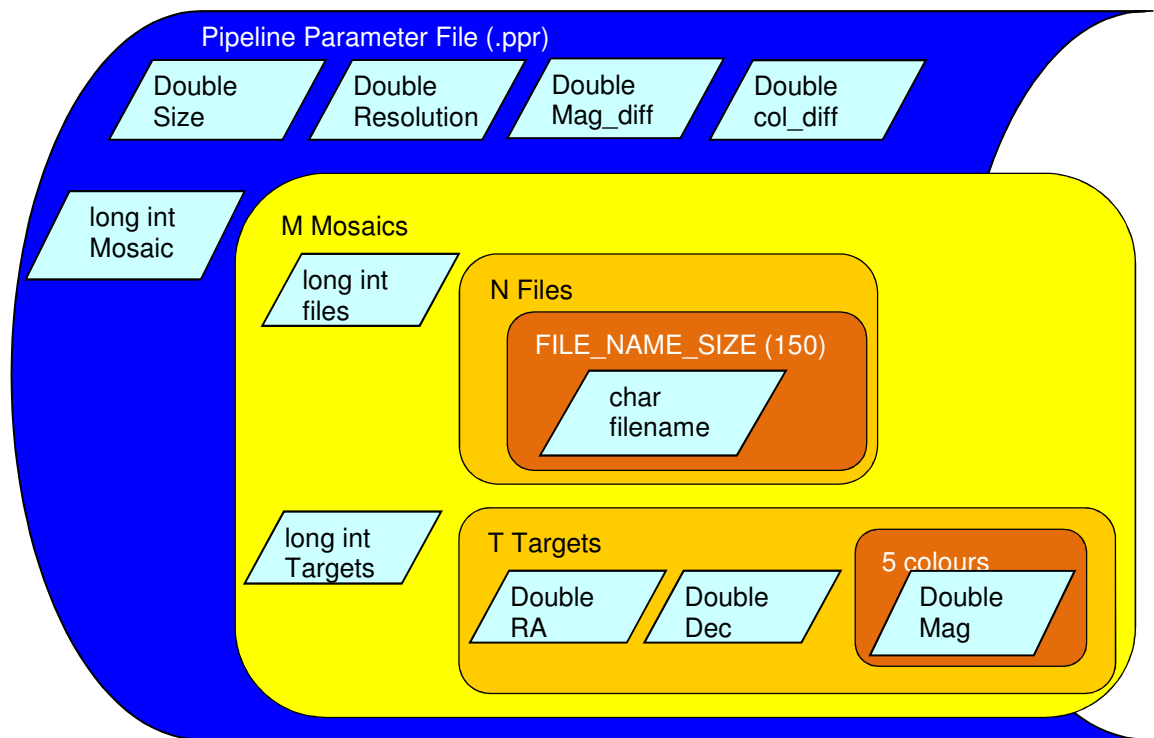


Figure 8-16 Pipeline Parameter Files (PPR): PPR files consists of 4 observational parameters, and a repeating structure for each mosaic to be generated, itself consisting of two repeating structures, one listing the paths to the files to be aggregated into that mosaic, and the other listing the targets to be analysed within that mosaic. Each repeating structure has a long int controlling variable.

8.3.3.3 Parameter Text Files (PRT)

BASH Shell scripts are used for grid job management in this project. Data from text files is piped into these scripts to parameterise them as discussed in Subsection 7.2.4. The PPR and PRM files are stored in binary format, which is not compatible for use with shell commands.

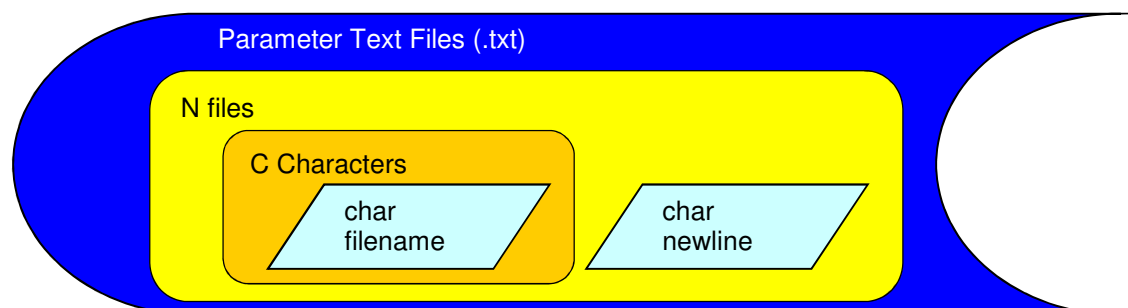


Figure 8-17 Parameter Text Files (PRT): these files are ASCII text files used to list file paths without requiring binary file access. They consist of a line for each file, terminated with a newline character, and each line consists of a sequence of characters representing the path to that file

In order to allow for the scripts used in the project to access the files listed in the parameter files, text versions of the Parameter files are generated in parallel with the binary parameter files listed above. These text files, contain lists of file paths, with each new path separated from the previous one by a newline character.

These files are referred to as PRT files in this project, to distinguish them from standard text files, although the extension used was `.txt`.

8.3.4. Text (TXT)

Text files are files which contain a sequence of zero or more characters in some form of encoding, [165] most commonly ASCII or Unicode. Text files are used for several purposes in this project.

Firstly, text files can be readily viewed using text editors on most common platforms, which makes them suitable for storing data in a user-accessible manner. Log files storing output from the project at various points are stored in this way.

The Windows and Unix platforms used in this project use different conventions for the end-of-line character. Utilities called “dos2unix” and “unix2dos” are used to convert files from one to another as needed. [166]

Secondly, scripts, program source code and header files are created and encoded as text files. The contents of these files are discussed under Software Design in Chapter 7.

Finally, some of the programs used in this project, such as the grid management shell scripts discussed in Subsection 7.2.4, use text- rather than binary-encoded files for I/O. These files can be viewed by the user with a text editor, but are more usually used in an automated way within the project. Three text file types are used for this purpose: Parameter Text files (PRT, Subsection 8.3.3.3) Comma Separated Variable files (CSV, Subsection 8.3.5) and Job Description Language files (JDL, Subsection 8.3.6)

8.3.5. Comma Separated Value (CSV)

Comma Separated Value (extension `.csv`) files are two dimensional tables of any size where the entries and the table structure are stored as text. Each line of the text file is a new line in the table and some delimiter, often a comma, is used to separate columns within each line. Both numerical and non-numeric data can be stored in this way. A common, but not universal, convention is that the first row of a CSV file is the “header” and includes the names of each of the columns, also separated by the standard delimiter. [163]

An advantage of this structure is that CSV files, as with plain text, can be accessed and edited using standard text editors. In addition, CSV files can be accessed by shell scripts.

A disadvantage of CSV is observed when it is used for storing numerical data. Numerical data is typically presented in the form of a decimal representation of the number in ASCII characters. As such, each character is a byte of data, and the higher precision to which a number is written, the more data storage is used. In addition, a character is used to represent the decimal point, and leading or trailing zeroes also require storage. This is in contrast with binary floating point numbers, which require a constant amount of data storage.

Throughout this project, CSV files are used to transfer data from the CAS to other computing resources used in the project. These files are used to reference the positions of targets and files used in the project. Each file uses the header convention explained above, which is used as a check to ensure that the CSV file submitted is the correct type. In the diagrams illustrating these file types, the headers are not shown, but are implied to be present unless otherwise specified

8.3.5.1 API CSV files

As part of the SDSS API, CSV files from the CAS are used as an input to the program `parameterise` as discussed in Subsection 7.2.3.1. These CSV files contain lists of `run`, `rerun`, `camcol` and `field` for each field in SDSS. They are used by the parameterisation software to create batches of PRM files. (see Subsection 8.3.3)

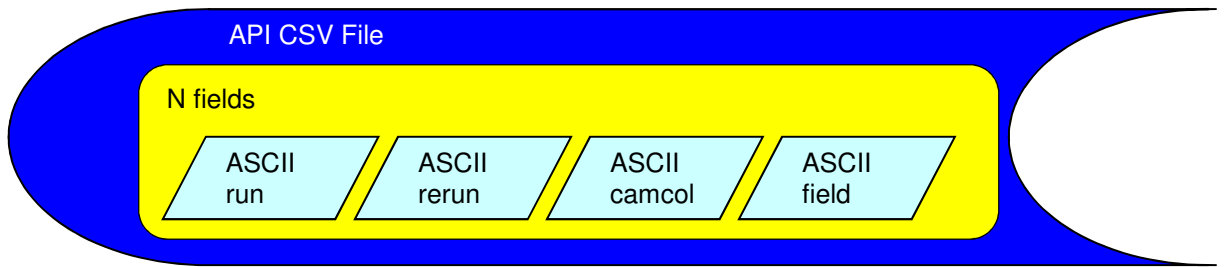


Figure 8-18: API CSV File: These files consist of lists of run, rerun, camcol and field for each of the fields to be processed. These numbers can be translated into file paths using the definition shown in Figure 8-8

8.3.5.2 Target List Pipeline

As discussed in Subsection 7.2.2, the pipeline is used in two different ways. The first method is to pre-select a number of individual targets to be processed. (e.g. the SDSS Quasar List) SQL queries to the CAS are used to find the fields within a FoV of those targets.

This generates a CSV list of targets as well as a list of the associated fields needed to make a mosaic around them. This CSV list is used as an input to parameterise to create PPR files.

Each line of the CSV file has:

- order information, consisting of
 - objectID, a unique identifier for each target
 - the number of fields that is needed to make the mosaic around that object, which is used with an index to control the loop in the parameterisation program (see Subsection 7.2.3).
- field information of one of the fields that are used to create the mosaic.
 - run
 - rerun
 - camcol
 - field
- target information pertaining to a specific target
 - RA
 - Dec

- Magnitudes (u , g , r , i , and z).

Once the index of the fields used to make the mosaic around a given target reaches the number of fields, the next mosaic starts. The structure of each mosaic is of a series of lines in the file, delimited by change in objectID as illustrated in Figure 8-19

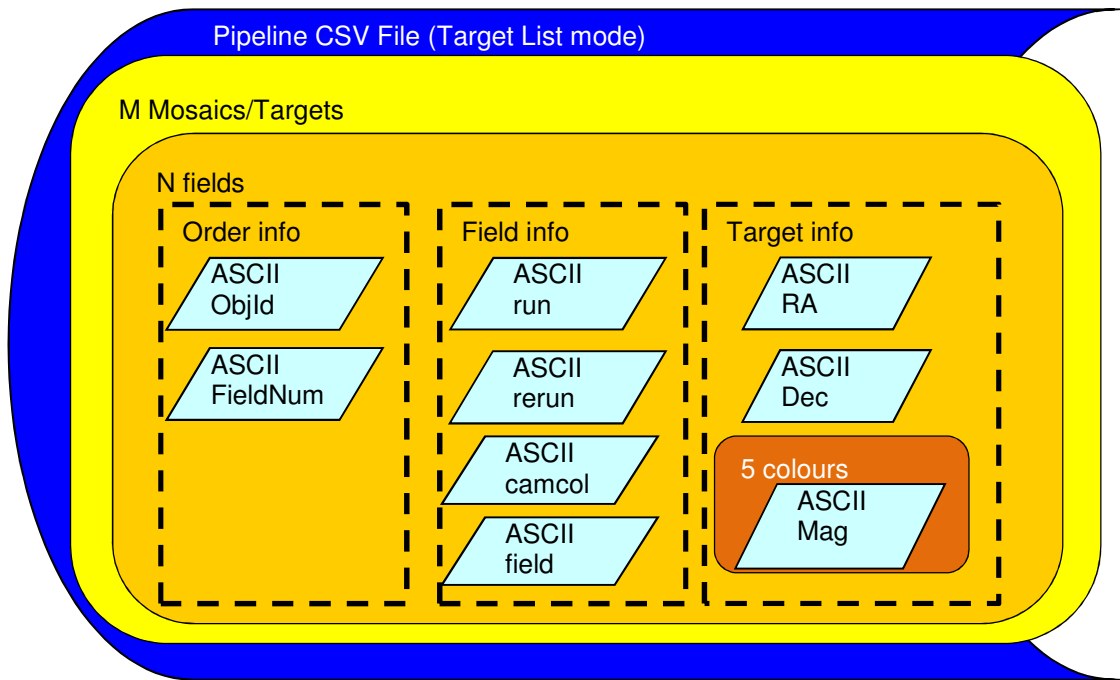


Figure 8-19 Pipeline CSV File (Target List mode): These files consist of a number of sets of lines, each corresponding to the mosaic of fields around a target in the target list. Within these groups, each line corresponds to a field in the mosaic. Target information is repeated on each row within the group.

8.3.5.3 Catalogue Traversal Pipeline

The second method of using the pipeline is to use the catalogue itself as a target list. Again, the CAS is used to find the files, but this time each field is associated with those fields which are within a FoV of *any* point of the target field.

This generates a CSV file containing a list of field descriptors for the target fields, as well as a list of the associated fields needed to make a mosaic around them. This CSV file is used as input to the parameterisation program to create PPR files as discussed in Subsection 7.2.3.3.

As before, each target field is referenced on multiple rows, with each of its neighbouring fields listed on those rows. Again, the number of rows is used as a control for the loop. In this method *all* targets within the fields specified in the CSV and PPR files are considered potential targets to be processed.

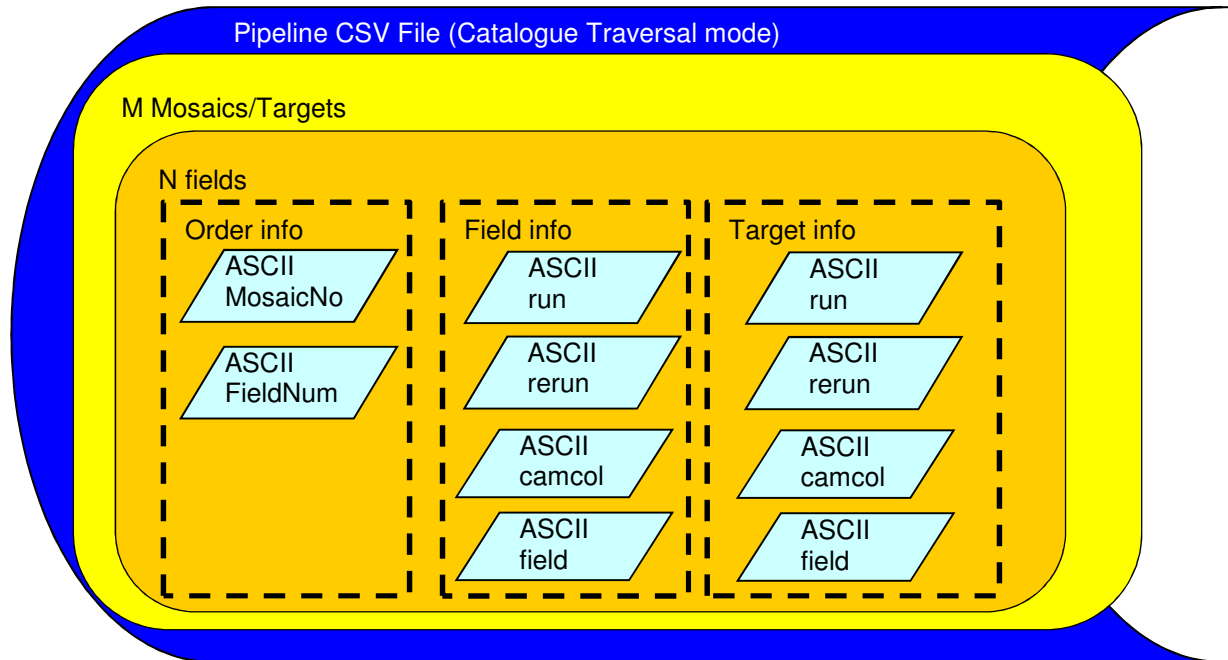


Figure 8-20: Pipeline CSV File (Catalogue Traversal mode): These files consisted of a number of sets of lines, each corresponding to the mosaic of fields around a target field. Within these groups, each line corresponded to a field in the mosaic. Target field information was repeated on each row within group.

8.3.6. Job Description Language (JDL)

The Job Description Language is the means by which jobs are submitted to the grid in the gLite system. [7] It works by specifying a number of attributes for a job. The attributes of a job are specified as a series of key-value pairs, where the key is selected from a pre-defined list of grid job attributes which can be specified, and the value is one or more strings specifying that attribute for that specific grid job. [7]

These attributes include the executable to be run, any arguments to that executable, and any files to be copied to the grid node from gridUI when the job is submitted. In addition, any further requirements may be specified for the job, such as a required amount of disk space, or a specific grid queue to submit the job to. All jobs for this

project are automatically generated by Bash shell scripts, as described in Subsection 7.2.4.

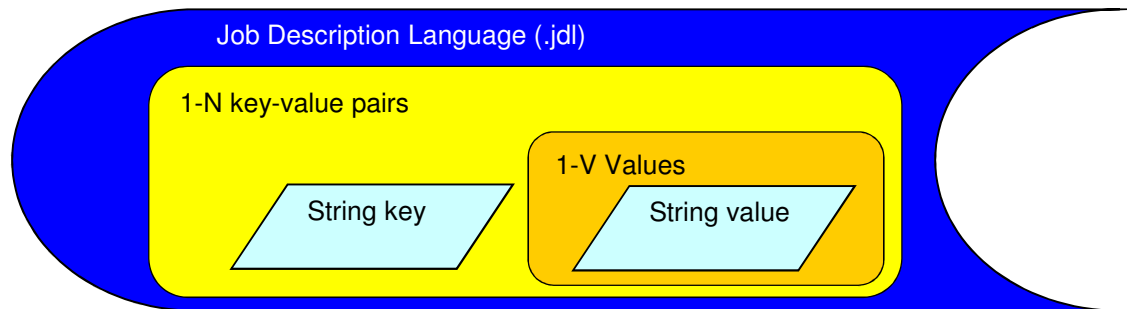


Figure 8-21: Job Description Language (JDL) files: These files consist of a number of key-value pairs specified in strings with specific formatting determined by the `gLite` User Guide [7].

Each JDL file describes a single job, and as such, each JDL refers to one set of data to be processed. In this project, PRM and PPR files describe a given set of data, so each JDL refers to a corresponding PPR or PRM file as arguments to the executable. In addition, some jobs require more arguments, which are entered by the user when the JDL is generated.

8.4. Summary

This project uses input data from diverse sources, and stores and processes that data on a variety of computing elements, and generates output in a selection of formats. In order to maintain control over the data and computing elements, this data is managed by identifying firm communications links between data and computing elements of the project, a well-defined data directory structure and a clear understanding of the internal structure of the file types used in the project.

Five primary data storage and computing elements were used in the project:

- The Development Environment, a virtual machine used for software development
- gridUI, a gateway machine used to access the grid infrastructure
- the GMS, which forms the backbone of the grid system
- the LFC, the large-scale distributed data storage system used to store catalogues

- Worker Nodes, the individual components of Grid Ireland which are used to execute individual grid jobs.

The data stored on these data elements is stored in a directory structure which is intended to maximise the portability of the data by ensuring that the data is stored in the same relative path on all storage elements. As data is needed for a job, test or other purpose, it is copied from the LFC to the corresponding location relative to the working directory of the data element that job is operating from.

The top levels of the directory structure are designed to enable expansion within the SCG beyond the scope of this project. Within the scope of this project, however, these levels are static at `ittd/grid_cdm/`. This level forms the primary level for this project. The principal subdirectories are listed below.

- Workspace: a set of directories in which source code for the C programs is stored
- Scripts: a repository for the BASH shell scripts which are used primarily for grid job management
- Test: a designated location where software testing is carried out with minimal impact on other software
- Data: the storage for source, local and output data catalogues
- Release: the folder for storage of the final versions of the software prior to submission for grid jobs

The data within these directories is structured in a variety of formats. The six formats of note were:

- The Flexible Image Transfer System (FITS), an established standard in the Astronomical community [164]
- The Catalogue Information Format (CTI) an internally defined format which is used to identify the contents of FITS files
- Parameter files (PRM, PPR, and PRT), an internally defined set of related formats, each of which specifies data required for a particular grid job
- Unstructured Text files
- Comma Separated Value (CSV) files, a portable file format used to store tabular data in pure text format. [163]

- Job Description Language files (JDL) the standard format used to submit grid jobs in the `gLite` system. [7]

This data management process allows for robust access to the data from a variety of data storage elements, extensible systems suitable for use with future projects and a flexible system which enables large-scale data to be accessible without requiring that each grid job have full access to the stored catalogue data.

9. Implementation & Operations

This Chapter discusses the pragmatic methodology used to bring the project from design to completion. Two key areas of operation are identified: the project implementation process and the practicalities of grid operations.

The discussion of the project implementation in Section 9.1 begins by describing the development environment in detail, with explanations of the effects this has on testing and implementation phases of the project. The development testing cycle is then discussed, emphasising the specific implementation of the cyclical design process illustrated in Subsection 6.2.1.

Section 9.2 discusses the practical approach needed in dealing with a grid. Hardware and software limitations on the grid which were encountered during this project are discussed in detail. These limitations demand a grid management process which is explained here.

9.1. Project Implementation

The process by which the project is transformed from the designs contained in Chapters 5-8 into the practical tool that produces results like those contained in Chapters 10-13 demands two elements.

The first is the development environment. Hardware resources available to the SCG dictate the nature of the development environment. Subsection 9.1.1 explains the nature of that environment and the restrictions this places on the project.

The project makes use of a variant of the Iterative and Incremental development method, [150] where design specifics of the project evolve over time as part of a development cycle discussed in Subsection 6.2.1. A critical element of this development method is testing, and the various testing levels and cycles used in this project are discussed in Subsection 9.1.2.

9.1.1. Development Environment

The primary development environment used throughout this project is a Virtual Machine (VM). This VM was initially operated through Sun Virtual Box version 1.6.

Successive upgrades were used, and the version used at the completion of the development phase was Oracle VM VirtualBox 4.2. This environment was initially used on a Windows XP desktop computer at ITTD, which was successively replaced with several newer model computers. The current environment is held within a laptop running Windows 8.

The use of a VM permits the transfer of the development environment wholesale from computer to computer over time as the project continues and hardware resources available to the project change. This ensures continuity of service and a stable environment for testing the project. In addition, the entire VM, 8GB in size, is backed up to external physical and cloud storage for additional reliability.

The operating system of the development environment is Scientific Linux version 4.5. The choice of Scientific Linux version 4.5 is dictated by the requirements of Grid Ireland – software is thus developed in a compatible environment with the grid.

The Integrated Development Environment (IDE) chosen for the project is Eclipse Platform version 3.3.1.1. This was selected on the recommendation of Tadhg O Briain [167] as a suitable environment for developing C and C++ programs with user-friendly debugging and compilation processes.

The use of virtualisation allows for shared directories to be established between the Linux virtual machine and the Windows host machine, enabling intuitive transfer of data from one to another for test purposes. However, the shared folder causes some problems when used on a host with 64-bit architecture while the VM is configured to simulate 32-bit architecture. For example, files opened by the project programs within the shared directory can become corrupted, and some programs may crash after 1023 lines of output are written to a file. This is corrected by ensuring test programs are run entirely within the virtual environment. From there they can later be copied to the host for further tests as needed.

9.1.2. Software Testing

Four major levels of testing are used over the course of the project, and each set of tests can be used to build upon the level below it. Firstly, functional testing uses dummy data within the IDE debugging system to test the operation of a particular function or

group of functions. This technique is useful to evaluate whether a function was operational or not, and can be used to highlight basic code bugs, but is not ideal for identifying design flaws and problems pertaining to scalability.

Second is small unit testing, which consists of running a project program on a truncated subset of test data (usually <200 lines), usually within the IDE. This technique is useful for demonstrating larger scale issues with the program, while being at a scale that the results can be double-checked using more primitive methods: e.g. repeating the calculation of the score for a small, simulated field by hand. In addition, small unit tests are quick and can be iterated rapidly as part of the development cycle.

Third is full unit testing – typically a slower process than the smaller scale tests but constituting a more rigorous software test. Full tests are carried out with subsets of the data comparable to those which are encountered in the release version of the project. Full unit tests are able to reveal issues such as memory leaks or file I/O errors which only became apparent with larger datasets. Unit tests included a number of worked examples similar to that shown in Chapter 10, where each step was examined closely for any faults. Successful testing at this level usually indicates a program that is ready for release.

Grid tests are the final level of testing in this project. These tests have a slow response cycle – a minimum of five minutes and a maximum of three days. Grid tests are only used after all other tests are passed, thereby enabling them to reveal grid specific issues as discussed in Subsection 9.2. Grid testing requires specific grid management scripts to be designed and written (as discussed in Subsection 7.2.4) and act as both a test of these scripts and a test of other project software operating in a grid environment. Grid testing takes place at the second highest level of the project development cycle, after the “live” grid use of the project.

Once a program has passed all of these tests it is considered a *release* version and uploaded to the `release` directory in the LFC as per Subsection 8.2.6.

9.2. Practicalities of Grid Operation

Grid operations pose particular challenges not encountered in other computing paradigms. This Section describes the nature of these issues and the solutions used to

mitigate the effect of them. These challenges can be categorised under the three headings listed below.

Firstly, programs submitted to the grid cannot simply be executed from a GUI as they would in a non-distributed computing system. Instead, they must be submitted to the Grid Management System (GMS) using JDL.

Second, though Grid computing provides substantial computing power as described in Section 3.2, that power is not infinite. The limitations of that power, and the constraints thereby placed on this project are discussed in Subsection 9.2.2.

Finally, this project's unusual nature placed additional constraints on the retrieval of output data from the project. [134]

9.2.1. Grid Job Submission & Scripts

Instead of being executed from the command line, grid jobs are submitted to the GMS which allocates jobs to the nodes. These grid jobs are created, managed and submitted by scripts designed for the purpose. The design of these scripts is discussed in Subsection 7.2.4. In addition to the additional software design and development load, this submission protocol places the following limitations on the project.

9.2.1.1 Grid Job Runtime

Grid Jobs are limited to a maximum of three days runtime. [134] This places limits on how many targets can be processed as part of a single job. Experimental submissions of varying numbers of targets to the grid were used to evaluate the relationship between file I/O, processing and output time, and thus to estimate the number of targets that could reliably be processed per unit time.

Experimental results of unit testing which lead to the determination of this relationship are discussed in Section 13.2. Because the grid jobs in the Exoplanet Catalogue are selected by means of fields in the Local Catalogue, rather than by target, and Local Catalogue fields themselves have a variable number of entries, significant variation in the size of the grid jobs was observed in the generation of this catalogue.

As a practical solution, grid jobs are calibrated such that the expected processing time is 36 hours to permit a 100% margin for variance in processing time. Further discussion of the metrics used to estimate grid runtime may be found in Section 13.3.

9.2.1.2 Simultaneous Jobs

A grid is, as discussed in Subsection 3.2.1, a large, but not infinite, collection of computers. The number of nodes that form the grid therefore places an absolute limit on the number of jobs that can be running simultaneously, in this case 768. [93] However, unless the grid is made available to a particular project as a dedicated resource, grid management must set limits on the number of nodes that can be assigned to a particular user.

At peak use during the creation of the Exoplanet Catalogue, Grid Ireland made 400 processors available to this project. As a result, it was necessary to manage grid job submissions in order to prevent the system from exceeding this limit using the `submit_jobs` script defined in Subsection 7.2.4.

Submission of jobs to the GMS is estimated experimentally to take between 10 seconds and 1 minute depending on grid workload. Since this project, at peak load, submitted a total of 1,791 grid jobs, it was advised that jobs be submitted in batches of no more than 10 at a time, at intervals of 5 minutes, to prevent the GMS from being pushed beyond capacity. The `submit_jobs` script also managed the issue of simultaneous job submissions as discussed in Subsection 7.2.4.

9.2.2. Data Limitations of the Grid

Grid operations are limited by both hardware and software constraints imposed by the grid architecture. Some of these are well documented and can be flagged in advance. These were built into the design of the project as discussed in Subsection 7.1.2.2. Others did not become apparent until revealed in practice by the project. Several data-specific issues that were encountered in the process of running the project are discussed below.

9.2.2.1 LFC Access times

As discussed in Section 8.1, files used for long-term storage within the grid environment are stored in the LFC. Access to the LFC is provided using the `gLite` grid software. `gLite` incorporates security and authentication protocols which are necessary for secure grid operations. [7] These protocols cause two major issues with projects like this one which use large numbers of small files.

The first is that a user's grid security certificate expires after 12 hours, even during the execution of a 3 day grid job. Therefore the certificate must be renewed at least that often. A script was written to renew the security certificate every 10 hours to resolve that issue.

The second issue proved more detrimental to the project. Using `gLite` to access files in the LFC activates an authentication process for each individual file. [7] This authentication process has been experimentally estimated as taking between one and five seconds. For typical grid jobs with a small number of large input files, a process that requires a few seconds for authentication does not pose a significant issue.

However, for projects like this one, this authentication and access time become a dominant factor in grid job runtime as explained in Section 13.4. This issue was mitigated for this project when a Network File System (NFS) to which WNs had direct access to was installed on the Grid. The Local Catalogue was replicated on that system.

The grid management scripts were modified to access the NFS first, instead of the LFC, and copy the data from that system to local storage. No other modifications were needed to the data or software.

9.2.2.2 Data Corruption and Losses

The LFC and other Grid Ireland resources are not under the control of the SCG. Due to the scale of the LFC, it was not fully backed up. This meant that when a disc containing part of the Local Catalogue as stored on the LFC failed, that data could not be retrieved from backups. Since Local Catalogue files are derived from source files, and given an identical name with the suffix “_local” as defined in Subsection 8.3.1.2, it was possible to identify the lost files by comparing the source catalogue with what remained of the

LFC. Then, it was decided to re-run the API on those files in the source catalogue which were missing from the Local Catalogue.

On a number of occasions during the creation of the Exoplanet Catalogue, grid jobs failed to complete due to a variety of reasons (e.g. exceeding the grid job time limits or data storage capacity of the WN or GMS) and the data that they would have generated was lost. In this case, a pragmatic attitude was taken: 67,043,579 stars had been analysed and the data recovered, representing 78% of the Local Catalogue processed. It was decided that this represented a sufficient data product for this project.

9.2.3. Result Generation & Collation

Over the course of this project, 77,429 quasars and ~86,000,000 stars were analysed using the Locus Algorithm. For each, an attempt was made to give coordinates for the optimal pointing for a telescope for the purposes of differential photometry, and a score to represent how good that star is for that purpose. This process produces large catalogues as discussed in Chapter 11.

Standard grid output would take the form of a set of log files created from the standard output from the programs run as part of that grid. [7] For such a large output, this is not viable as discussed in Subsection 9.2.3.1. Instead, this project made use of the large storage facility available through the use of the LFC as explained in Subsection 9.2.3.2.

9.2.3.1 Logfile/Standard Grid Output

Previous practical experience in Grid Ireland involved the use of medium-large input datasets, large processing demands, and very small amounts of output data. [134] This meant that it was possible for programs running on the grid to send their output to `std.out`, which was piped by the GMS into log files, which could be retrieved using standard `gLite` commands. [7]

When this technique was tried with this project, it became necessary for management at Grid Ireland to perform an emergency shutdown on the grid jobs as they greatly exceeded the available storage space for the log files, and were disrupting grid availability for other users.

In addition, the log files are ASCII files, while output from the project is numeric. Therefore there was a risk of loss of precision in the output data which was unacceptable.

In order to resolve the issues whereby the output files were overwhelming the GLS, grid job scripts were split into verbose versions used for testing and silent versions used on the grid. Instead of the log files, output from the project was saved in FITS files on the LFC as defined in Subsections 8.3.1.3 and 9.2.3.2.

9.2.3.2 LFC Storage of Output

As explained in Subsection 8.3.1.3, output from this project is created in the worker nodes on the grid and then copied to the LFC. The high capacity of the LFC makes it an ideal environment for the storage of large files like those used to store the grid output, as discussed in Section 13.1. In addition, since each job produces a single file from many, the latency due to LFC access times as highlighted in Subsection 9.2.2.1 is not a significant cost on any particular job when used for storage of output.

9.3. Summary

Implementation of this project largely follows the design laid out in Chapters 5-8. The development environment, a virtual machine operating Scientific Linux version 4.5 enables portable development compatible with Grid Ireland requirements. By developing in this environment, which was operated through a variety of versions of VirtualBox on physical Windows devices, the project can have firm control over environmental parameters.

The testing process in this project is carried out at four primary levels.

- Functional testing, which tests functions and other low-level components
- Small Unit testing, which tests programs and higher level components for semantic errors
- Large Unit testing, which tests programs and higher level components for operational errors such as memory leaks and timing issues
- Grid testing, which tests high level components of the project for issues specific to the grid.

In addition, the full local, quasar and exoplanet catalogue generation systems can be considered *de facto* tests of the system as a whole as these jobs were monitored to show computing metrics for the overall system.

The implementation of this project also demanded a number of pragmatic decisions be taken regarding operations on Grid Ireland. Firstly, Grid job submission has to be monitored for a number of practical issues. The monitoring was required to:

- plan Grid jobs such that they do not exceed grid runtime limits. A margin of error of 100% on a runtime limit of 72 hours leads to a scheduled runtime limit of 36 hours.
- ensure that the number of jobs submitted did not exceed the maximum number of jobs permitted
- limit the number of jobs submitted simultaneously

Secondly, the Grid has several issues related to data access and reliability.

- Access to the LFC for many small files as used in this project is dominated by security and authentication procedures. The use of an NFS in place of the LFC allows for more rapid, reliable access to the data
- Complete backups of all storage were not made available through the LFC. Where possible, as with the Local Catalogue, data lost due to disc corruption was replaced. Given the data volume in this project, when this was not possible, it was determined that the remaining data constitutes a significant data product.

The production of large catalogues of output data as is the case with this project is not typical of prior Grid Ireland projects. As a result, several protocols have to be implemented to prevent disruption to the GMS.

- Operational programs and scripts suppress their output which would usually be piped to `std.out`. Specific verbose versions were developed as needed for testing purposes which do not suppress this output.
- Output data is exclusively stored in the LFC.

10. Individual Result

This Chapter provides a complete worked example of the project's software solution in operation on a single sample star, SDSS J113824.40+483457.8. This Chapter shows a step-by-step description of the process which was used to generate pointings for that star.

This same process was iterated 67,043,579 times to produce 61,662,376 pointings for other stars in the Exoplanet Catalogue, as discussed from a broader perspective in Section 11.4 and Chapter 12.

This particular star was selected as part of a follow-up observation series at Raheny Observatory (Minor Planet Centre reference MPC #J41) [168] in the Spring of 2014. SDSS J113824.40+483457.8 was selected from the catalogue on the following basis:

- It is sufficiently bright ($r = 15.7938$) that it was well within the capacity of the 0.35m telescope at Raheny to observe at high signal-to-noise ratio (SNR) with 1 minute exposures.
- Its position (RA 174.6017°, Dec 48.5828°) was visible above the useful visible horizon from early in the night from Raheny Observatory during the Spring of 2014
- Its score of 10.4877 was approximately equal to the median for stars in its magnitude range, and therefore constituted a typical example of a star in the range which the telescope at Raheny Observatory could observe. This score, was previously calculated as part of the Exoplanet Catalogue, as discussed in Section 11.4, using the method shown in Figure 5-3, from the reference stars shown in Table 10-1. This chapter provides a complete description of how this score was calculated.

In the following Sections the process by which this target was analysed in the catalogue creation phase of the project are repeated here and explained in detail. This process is logically partitioned into three Sections: extraction of data from SDSS, identification of candidate reference stars and application of the Locus Algorithm.

Finally, there is a brief discussion of an observing run of this target from Raheny Observatory based upon the pointing indicated by the Locus Algorithm analysis. Detailed analysis of the output from this observing run is in progress at time of writing.

10.1. Aggregate Data from SDSS

This first step shows how data for candidate reference stars was extracted from SDSS for use in building a mosaic of fields around the target in this worked example. This Section corresponds both to the API step of the project as described in Subsection 7.2.1 and the SQL query through the SDSS CAS which identifies fields within a FoV of a target.

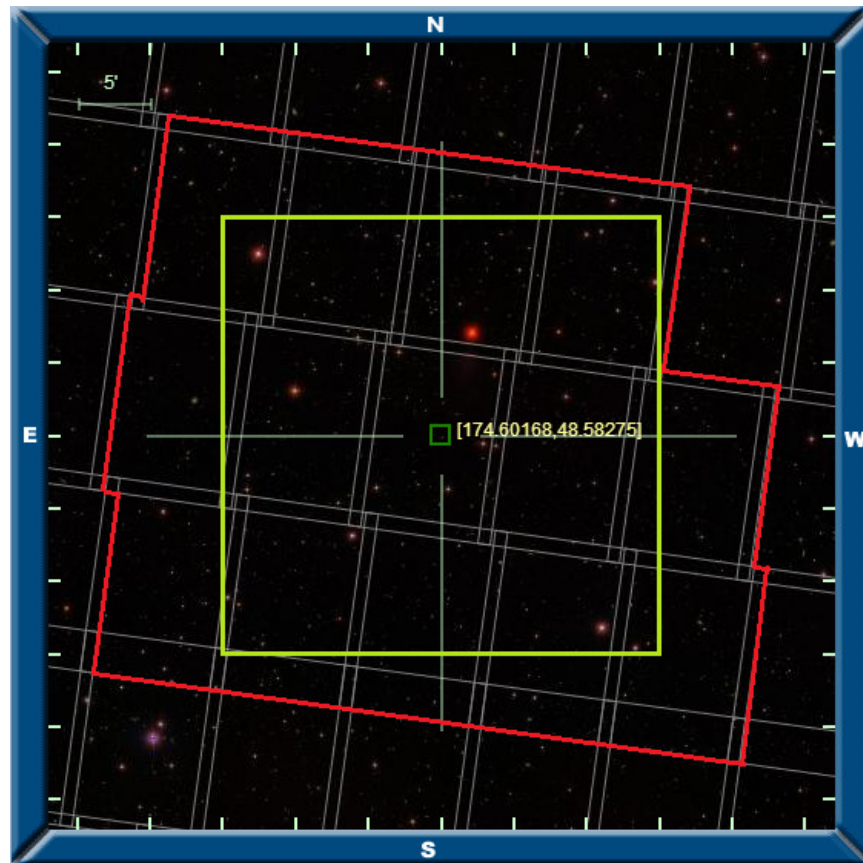


Figure 10-1 Fields required for target star SDSS J113824.40+483457.8 (dark green). All areas which can be included in a FOV with the target are highlighted in bright green. The 14 fields which must be accessed to create this area are highlighted in red. Image taken from SDSS Navigate tool. [140]

In the course of this project, a complete set of SDSS Calibrated Objects (`tsObj*.fit`) files was downloaded from SDSS. From these files the Local

Catalogue was extracted. The Local catalogue excludes any objects in SDSS other than stars, and for each of those stars only lists its position (RA and Dec) and magnitude (*ugriz*).

Each file in the local catalogue corresponds directly with a file in the SDSS source catalogue. As a result, it is possible to identify which fields in SDSS are required to show all stars that could potentially be used as a reference star for the target star. These fields were identified by means of an SQL box search query through the SDSS CAS as explained in Subsection 7.2.5. This query selected all distinct fields within 0.25° (the size of the FoV) of the target in Declination, and, after applying the correction factor defined in Section 5.1 (dividing 0.25° by the cosine of 48.5828°) within 0.3779° of the target in Right Ascension.

As can be seen from Figure 10-1, 14 SDSS fields have at least some component within these bounds. In the original SDSS files, there are 10,813 catalogue entries in these 14 fields. After the application of the API as shown in Subsection 7.2.1, the Local Catalogue versions of the same files contain 2,447 stars. See Subsection 11.2 for further analysis on the conversion between SDSS Source data and the Local Catalogue.

10.2. Identify Potential Reference Stars

This second step shows how all of the stars in the mosaic of fields close to the target are compared with the target, and only those stars which are suitable are considered as candidate reference stars. The suitability of these stars is determined by a series of filters which are based on user-defined parameters. These filtration steps are carried out by the Filter function which is part of the main data pipeline as shown in Subsection 7.2.2.1.1. Only the candidate reference stars which pass all of these filters are used in the application of the Locus Algorithm itself.

The data from the 14 fields on all 2447 stars was read into memory from the corresponding Local Catalogue files. This data was subjected to a series of filters to screen out stars which were unsuitable to be used as references for the target.

The first filter excluded any star which more than the size of the field of view away the target in North-South/East-West translation: i.e. it excludes any star North of Dec 48.8328° , South of Dec 48.3328° , West of RA 174.2238° or East of 174.9796° . 1457

stars remained that could be included in a FoV with SDSS J113824.40+483457.8, as shown in Figure 10-2.

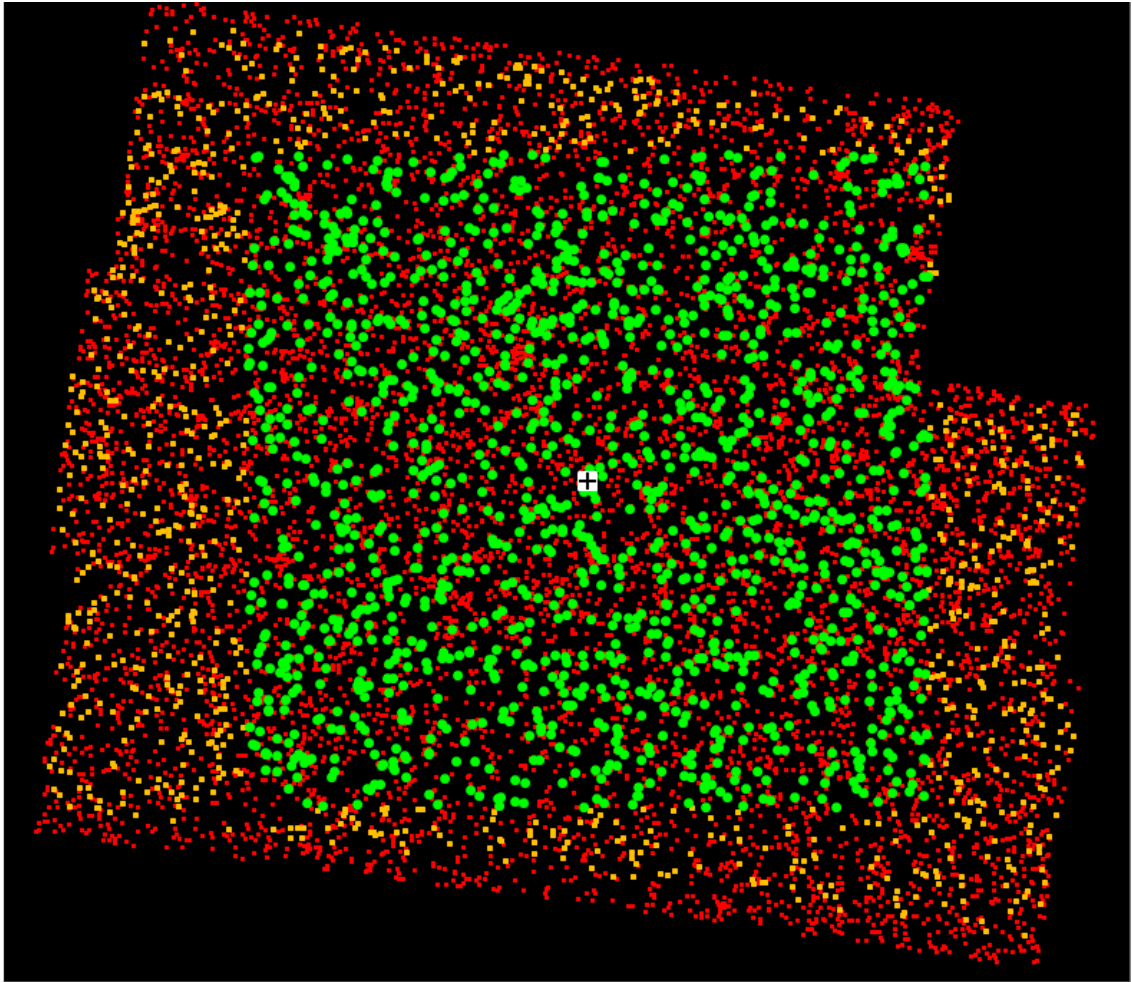


Figure 10-2, There are 10,813 entries (shown in red) in SDSS among the 14 fields required for SDSS J113824.40+483457.8. Of these, 2,447 met the criteria for a star (highlighted in orange.) Only the 1,457 stars marked in green could be included in a FoV with the target (white with black cross).

Second, to avoid field crowding, any stars which were too close to another star to be clearly resolved were eliminated. In this case, in consultation with BCO and Raheny Observatory, the resolution parameter was set at 1 arcsecond (0.0002777°.) [83] [23]

While this parameter is finer resolution than the seeing at either location would usually permit, this parameter was chosen as it is intended to use this system in conjunction with systems for Lucky Photometry as discussed in Subsection 2.2.3.4.2. Since Lucky Imaging systems have the potential to permit near-diffraction limited imaging for

modest telescopes, [81] 1 arcsecond was selected as a suitable value slightly larger than this limit. No star failed to meet this parameter.

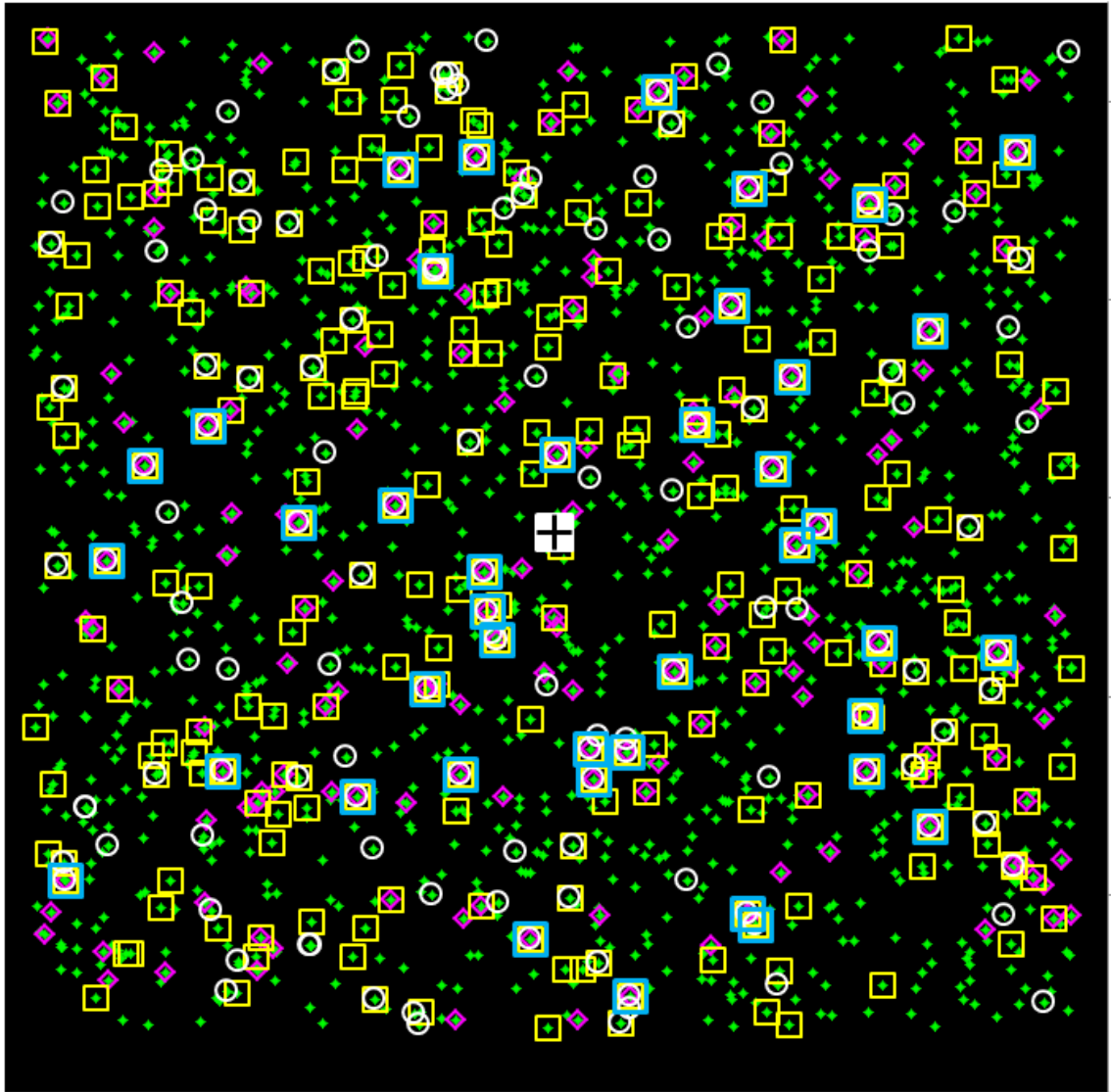


Figure 10-3: Filters to identify reference star candidates. The 42 candidate reference stars, highlighted in cyan boxes in the were selected from among all stars (green diamonds) within a FoV of the target (white and black cross) by including only those which were of the correct magnitude (white circles), and which matched the target on both $g-r$ (magenta diamonds) and $r-i$ (yellow squares) colour indices.

The next filter removed stars which are too bright or too faint for use with the selected target. Stars within ± 2 magnitudes in r of the target were passed to the next filter. This value was chosen in consultation with Niall Smith of CIT to ensure that the faintest stars observed were observed with high SNR while the brightest stars were not saturated.

[23] This meant that stars with $13.7938 \leq r \leq 17.7938$ were in the right magnitude range. A total of 143 stars met this criterion

Two more filters were applied, based on colour indices: targets with $g-r$ and $r-i$ within ± 0.1 mag of that of the target (0.4662 and 0.1606 respectively) were passed. This limit was selected to be well within the proposed colour variation proposed by Young (1991), which suggests a limit of 0.3 mag difference in Johnson B-V colour index. [51] 170 stars passed the $g-r$ filter and 273 stars had an $r-i$ within the correct range.

Once a potential reference star passed these three filters, a rating was calculated for that reference. This rating was composed of a triangular rating as discussed in Section 5.3.1 based on each of the two colour indices used above, i.e. $g-r$ and $r-i$. These ratings were combined by multiplying one by the other to form a final rating for each reference star. The formula used is given in Equation 5-4.

$$\begin{aligned}
C_{short} &= g - r = 15.2348 - 14.8277 = 0.4071 \\
C_{long} &= r - i = 14.8277 - 14.6985 = 0.1292 \\
\Delta C_{short} &= C_{Target} - C_{Reference} = 0.4662 - 0.4071 = 0.0591 \\
\Delta C_{long} &= C_{Target} - C_{Reference} = 0.1606 - 0.1291 = 0.0315 \\
R_{short} &= 1 - \left| \frac{\Delta C_{short}}{\Delta C_{max}} \right| = 1 - \left| \frac{0.0591}{0.1} \right| = 1 - 0.591 = 0.409 \\
R_{long} &= 1 - \left| \frac{\Delta C_{long}}{\Delta C_{max}} \right| = 1 - \left| \frac{0.0315}{0.1} \right| = 1 - 0.315 = 0.685 \\
R_{long} &= R_{short} \times R_{long} = 0.409 \times 0.685 = 0.280
\end{aligned}$$

Equation 10-1: Expansion of Equation 5-4 for one candidate reference star, SDSS J113749.38+482307.1 showing how its rating of 0.280 was calculated

As a worked example, this equation was applied to one arbitrarily chosen candidate reference star, SDSS J113749.38+482307.1. This star, located slightly south-west of the target at RA 174.4558°, Dec 48.3853° is slightly brighter than the target at $u=16.4119$, $g=15.2348$, $r=14.8277$, $i=14.6986$ and $z=14.6667$, and passed all three filtering criteria. Equation 10-1 shows a step-by-step calculation of the rating of this reference star. This process, applied to each reference, was used to form the basis for the scores for each of the potential pointings in the Locus Algorithm.

Combining these three filters created a list of 42 potential reference stars – each of which can be included in a field of view with the target, is not too close to another star, is similarly bright, and of very similar colour to the target. This selection process is illustrated in Figure 10-3.

10.3. Apply the Locus Algorithm to each Candidate

The third and final stage of the process to identify an optimised pointing for differential photometry observation demonstrates the application of the Locus Algorithm to a set of candidate reference stars. This step corresponds to the Locus Main function, part of the data analysis pipeline as defined in Subsection 7.2.2.1.2

The Locus algorithm was applied to the set of 42 candidate reference stars to identify the optimum pointing for SDSS J113824.40+483457.8. For each candidate reference star, a boundary box was drawn, indicating the locus of points upon which the centre of the field of view could be placed to include both the target and that reference.

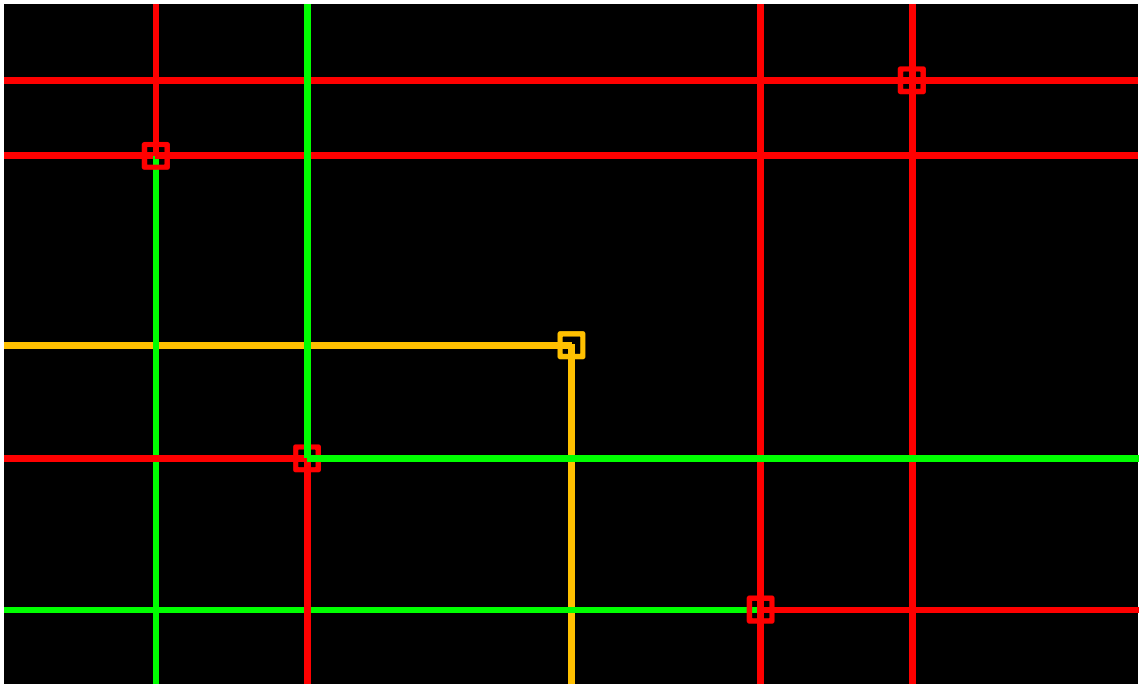


Figure 10-4: Possible Intercepts. For a given corner point and locus (yellow) there are four relative positions other cornerpoints can reside at. From each of these, Loci can be drawn North or South and East or West. Loci that produce an intercept are shown in green, those that do not are shown in red.

This boundary box was defined by a cornerpoint and a pair of Boolean variables as discussed in Subsection 5.2.3. The cornerpoint is defined to be exactly half the size of the field of view (i.e. 0.1250 degrees in Dec and 0.1890 degrees in RA) away from the reference star in the direction of the target. The Boolean variables indicated the direction from that point, along which a line was traced, North or South and another line traced East or West, back in the direction of the candidate reference star to give the correct locus of points.

In the case of the sample candidate reference star, SDSS J113749.38+482307.1, the boundary box was defined by the cornerpoint at RA 174.6447 and Dec 48.5103. The locus lines for this reference were drawn to the West and the South from this point.

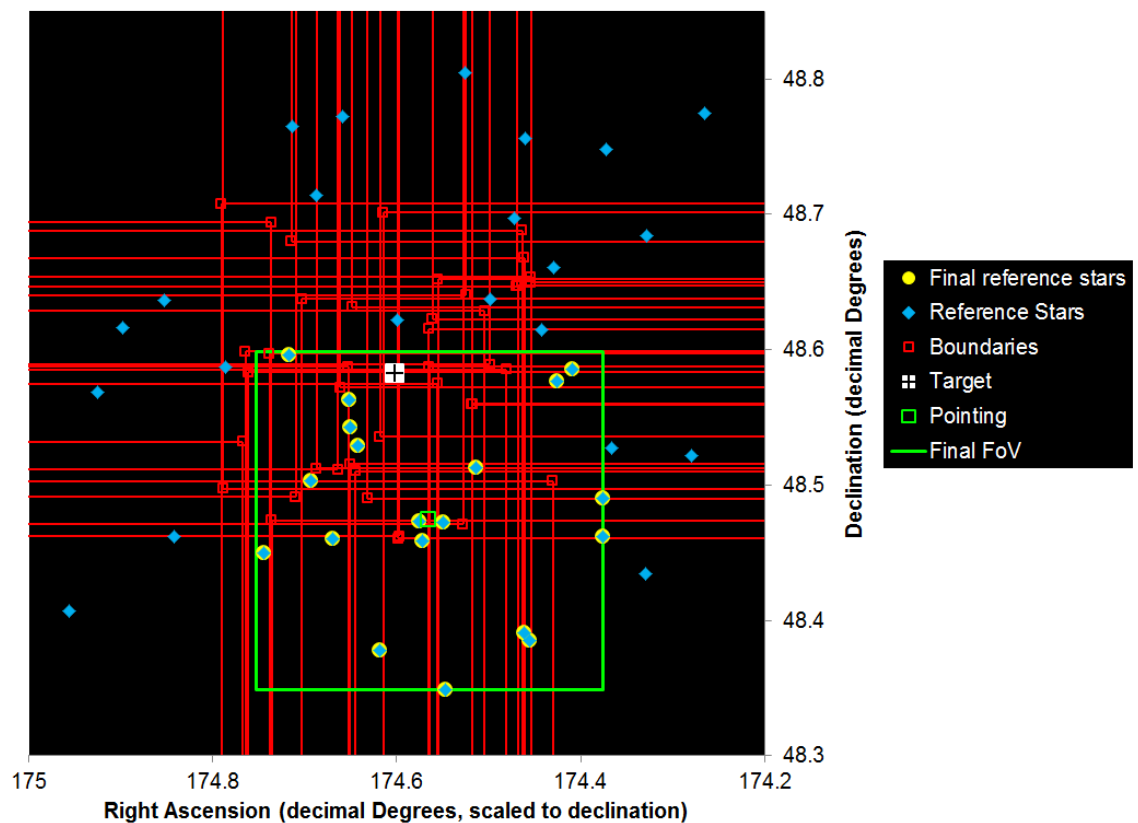


Figure 10-5 The Locus algorithm applied to all 42 candidate reference stars, indicated by blue diamonds. The Loci for each of the references are shown in red. The optimum pointing, and the Field of View centred on that point are highlighted in Green. The reference stars used in this optimum pointing are highlighted with yellow circles.

The repetition of this process for each of the candidate reference stars produced the pattern of intersecting boundaries shown in Figure 10-5. As shown in Figure 5-1, the

score for a given pointing changed only at the interception points between these boundaries. As described in Subsection 5.2.4, these interception points were defined by the RA coordinate of one cornerpoint and the Dec coordinate of another. Whether intersection existed between two boundary boxes was checked by the `set_boundary` function, and was defined by the direction the boundary lines were drawn from those points as illustrated in Figure 10-4.

In the case of the sample reference star SDSS J113749.38+482307.1, its boundary box had a cornerpoint at RA 174.6447, Dec 48.5103 and boundary loci drawn to the West and South. There were four possible cases when this locus was compared with the locus about another reference star as described below and illustrated in Figure 10-4.

- No intercept existed for any boundary box whose cornerpoint lay to the North-East of its cornerpoint.
- An intercept existed for any cornerpoint to the North-West of the sample cornerpoint, *if* the North/South locus from that cornerpoint was drawn to the South. That intercept lay at the RA of the other boundary box and the Dec of the sample.
- An intercept existed for any cornerpoint to the South-East of the sample cornerpoint, *if* the East/West locus from that cornerpoint was drawn to the West. That intercept lay at the RA of the sample and the Dec of the other boundary box.
- An intercept existed for any cornerpoint to the South-West of the sample cornerpoint, *if* the loci from that cornerpoint were drawn to the North or East. If the loci were drawn both North and East, two intercepts existed. The former intercept lay at the RA of the other boundary box and the Dec of the sample. The latter intercept lay at the RA of the sample and the Dec of the other boundary box.

The process of identifying intercepts reduced the number of times the scoring process had to be applied from a potential maximum of 1936 (42^2) to 947. At each valid intercept, a score was calculated by adding together the rating for each candidate reference star within half the size of the FoV of the intercept.

At each of the 947 iterations of this process, the score for that point was compared with the highest score achieved for any pointing for this target so far. If the score was higher than the best one so far, the current score was retained, otherwise it was discarded and the process repeated for the next valid intercept.

Once this process was completed, the pointing shown in Figure 10-5 was identified. That pointing lies at RA 174.5643°, Dec 48.4739°. The score associated with that pointing is 10.4877. This score was calculated by combining the ratings for a total of 20 reference stars around that point. The positions, magnitudes (*ugriz*) and ratings of each of these reference stars are shown in Table 10-1. The target itself is listed among the references.

RA	Dec	<i>u</i>	<i>g</i>	<i>r</i>	<i>i</i>	<i>z</i>	rating
174.3753	48.4623	15.7857	14.4585	14.0245	13.8799	13.8701	0.5697
174.3762	48.4905	18.4131	17.2971	16.9297	16.8015	16.7790	0.0079
174.4096	48.5857	16.1848	14.9651	14.5601	14.4239	14.4252	0.2935
174.4258	48.5764	17.6403	16.2228	15.7365	15.5762	15.5623	0.7975
174.4558	48.3853	16.4119	15.2348	14.8277	14.6986	14.6667	0.2800
174.4616	48.3907	19.1028	17.9124	17.4305	17.2351	17.1800	0.5502
174.5140	48.5127	17.2860	16.1655	15.7987	15.6661	15.6576	0.0049
174.5468	48.3489	18.6297	17.6274	17.1599	16.9873	16.9563	0.8691
174.5491	48.4722	17.0500	15.7185	15.2597	15.1151	15.0624	0.7775
174.5723	48.4585	16.1364	14.7226	14.2351	14.0891	14.0658	0.6727
174.5753	48.4736	19.0045	17.7761	17.2607	17.0743	16.9931	0.3776
174.6017	48.5828	17.4388	16.2600	15.7938	15.6332	15.5839	1.0000
174.6190	48.3779	17.0760	16.1191	15.6677	15.4716	15.4082	0.5490
174.6427	48.5287	15.7352	14.6496	14.2462	14.1495	14.1076	0.1343
174.6502	48.5429	17.0786	15.9560	15.5713	15.4439	15.4197	0.1236
174.6523	48.5631	17.9506	16.7423	16.2981	16.1474	16.1058	0.7021
174.6699	48.4605	16.5431	15.2885	14.8413	14.6840	14.6431	0.7832
174.6934	48.5036	18.1210	16.9843	16.5839	16.4509	16.4315	0.2478
174.7168	48.5963	19.0405	17.8163	17.3542	17.1977	17.1262	0.9195
174.7443	48.4498	17.5230	16.2742	15.8235	15.6608	15.6258	0.8276

Table 10-1: Table of reference stars for SDSS J113824.40+483457.8 (highlighted in yellow.) One of the reference stars, SDSS J113749.38+482307.1, is shown in red and is used to provide a worked example of the rating and scoring system

10.4. Observation of a Pointing

The results of this analysis were used to guide a series of observations. These observations were made at Raheny Observatory using a 0.35m telescope and a linear CCD camera (Kodak model SBIG-ST8XME.) This camera has an array of 1530 × 1020 pixels. The plate scale is 1.33arcseconds per pixel. [169] This gives a rectangular FoV of 34 × 22.67 arcminutes, substantially larger than the 15 arcminute square FoV for

which the Exoplanet Catalogue analysis was originally designed. This permits exploration of the effect of larger or smaller FoV on observation as discussed in Subsection 14.1.3, by artificially increasing or reducing the FoV used.

On the night of 15-16th February, 2014, a series of 269 images were taken from Raheny Observatory, each with a 60s exposure. A full suite of calibration frames were also provided in the form of Dark Frames, Bias Frames and Flat Frames. A sample of one of these images is shown in Figure 10-6.

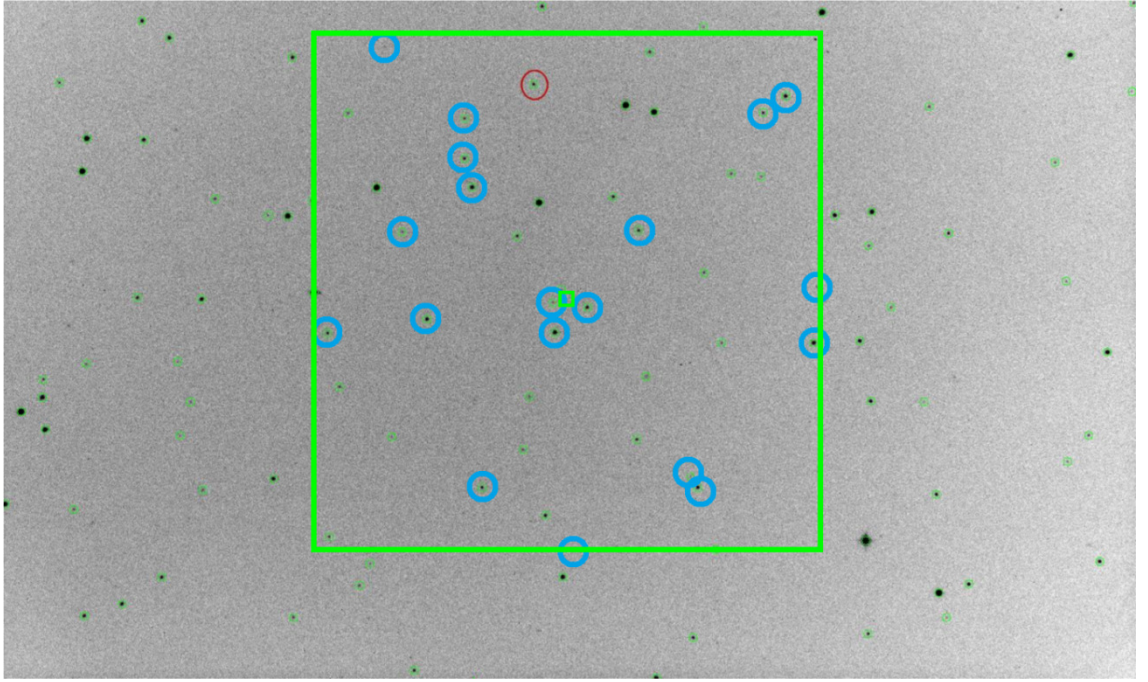


Figure 10-6: Image taken from Raheny Observatory based on pointing for SDSS J113824.40+483457.8 (circled in red). Reference stars are circled in blue. Note that the FoV for Raheny Observatory is larger than the FoV for which the Exoplanet catalogue was originally intended. The size and position of the original FoV and pointing is approximated in a green overlay.

A follow-on project is currently underway to attempt to use the images produced by this observation to produce differential photometry lightcurves using the reference stars shown. Further refinement to these lightcurves is expected to be possible by a series of experiments as discussed in Subsection 14.3.2.1

10.5. Expansion to the Catalogue on the Grid

The process shown in this Chapter represents one iteration of the full data analysis process, from data access to SDSS to the generation of an optimised pointing from

which observations can be planned and carried out. This iteration, however, represents just one target star, in one field, in one job as part of the generation of the Exoplanet Catalogue.

The Exoplanet Catalogue was generated using the same SDSS band (r), FoV (0.25°), resolution (0.0002777°) and magnitude- (2.0) and colour-limit (0.1) arguments as were used in this sample entry.

Each grid job consisted of identifying the pointings for each star in a collection of fields in the Local Catalogue. These grid jobs were each represented by a JDL file which defined the requirements of the job as shown in Subsection 8.3.6. A total of 1,791 such grid jobs were submitted, of which 1,598 successfully produced output files.

For each of 200 fields in every grid job, a new mosaic was generated as shown in Section 10.1: an SQL query to the CAS identified the files needed from the Local Catalogue to generate the mosaic around that field. These fields were then aggregated into an array in memory as shown in Subsection 5.2.1.

Fields in the Local Catalogue ranged in size from 0 to 4,541 targets, with a mean of 420. For each target in a field, a new list of candidate reference stars was generated as shown in Section 10.2. Those reference stars are used to generate each new pointing as illustrated in Section 10.3.

The final Exoplanet Catalogue, as discussed in Chapters 11 and 12, consisted of 67,043,579 entries for targets which had been processed. Of these entries, optimum pointings have been generated for 61,662,376.

10.6. Conclusions

This Chapter has shown the process by which a pointing can be generated for a given sample star, SDSS J113824.40+483457.8. First, the SDSS fields which contain nearby stars must be identified – in this case, there are 14 such fields. The Local Catalogue files, containing only stars, corresponding to fields must be read into memory – 2447 stars are listed in those 14 files.

A series of five filters are applied to remove stars which are unsuitable as reference stars: two correspond to position: one eliminates stars which cannot possibly be

included in a FoV with the target, while the other eliminates those which cannot be resolved from one another. One filter corresponds to magnitude, excluding any stars much brighter or much fainter than the target. Finally, there are two filters based on colour indices, which exclude stars which do not closely match the colour index of the target. Taken together, these five filters left 42 candidate reference stars.

Each candidate reference star is given a rating, and a boundary box is drawn about it. The rating process was explained with reference to a sample candidate reference star, SDSS J113749.38+482307.1. The boundary box is defined by a cornerpoint – literally the corner of that box, and a pair of binary switches indicating which direction from the cornerpoint a corresponding pair of lines should be drawn to complete the box.

The points of intersection between these boxes are identified mathematically and, where they intersect, a score is calculated by adding together the ratings for each candidate reference star that could be included in a FoV centred on that point. The point with the highest score is designated the optimum pointing for that target for the given parameters.

This pointing can be used to guide observations, such as the sample observation carried out at Raheny Observatory on the night of the 15-16th February, 2014. These observations can be used to expand upon the work of the project as discussed in Subsection 14.3.2.1.

11. Catalogue Outputs & Analysis

This Chapter presents the main results of the project: the catalogues generated and the analysis of the contents of one of those catalogues.

Section 11.1 first discusses what the catalogues consist of, in terms of the data contained within them, as well as an explanation of the meaning of that data. It then explains the value added to the data from SDSS by the generation of these catalogues.

Section 11.2 discusses the Local Catalogue. First, it discusses the generation and contents of the local catalogue. The variance introduced by the use of the reduced version of the clean sample of point stars algorithm in the generation of the Local Catalogue is explained in this Section.

Sections 11.3 and 11.4 respectively discuss the Quasar and Exoplanet catalogues. These two Sections first explain the inputs used to generate the catalogues, then give a high level overview of the data processing and outputs of that catalogue, and finally explain the purpose of their respective catalogues, both in terms of the role served within the project, and the results generated in the astronomical context.

A phenomenological meta-analysis of the output data for the Exoplanet catalogue is discussed in detail in Chapter 12. This analysis is intended to characterise the relationship between input parameters such as magnitude and colour index and output score. In addition, by providing detailed descriptive statistics of the output it allows users to better interpret it.

11.1. Overview of the Catalogues Generated

The Catalogues presented in this Chapter form the primary output from this project. These catalogues take the form of one or more FITS table files each with zero or more rows, each with information pertaining to a target. The catalogues present, for each target, its position in RA and Dec (stored as double-precision floating point numbers) and its magnitude in each of the five SDSS bands (stored as a vector of 5 floating point numbers.)

For a given set of input criteria as described in Subsection 7.2.2, the output catalogues also present (RA, Dec) coordinates and a score for the optimised pointing, all stored as double precision floating point numbers. In the catalogues generated by this project, as in SDSS, RA and Dec are quoted in decimal degrees.

The score quoted here reflects the degree to which the target and the sky around it meet the observational criteria set by the user, as calculated by the Locus Algorithm (See Sections 5.3 and 10.3.) These scores can be compared to one another: a target with a higher score meets the criteria better than one with a lower score.

As discussed in Subsection 5.3.2, these scores are intended to reflect the degree to which the sky around a target is suitable for differential photometry by providing a combined measure of the number and quality of reference stars located nearby. In the additive system each point of score marks one “perfect” reference star, or a combination of less closely matched reference stars whose scores add up to 1. Subsection 14.3.2 discusses in detail ways in which these criteria can be refined.

This comparison is intended to enable a user to determine which of a number of otherwise similar targets would be most suitable for precise observation using differential photometry. By this means, a user can plan an automated survey by preferentially targeting objects from which the most precise photometry can be obtained.

Subsection 11.3 describes a catalogue generated for all quasars in SDSS based upon criteria determined by observational experts at Blackrock Castle Observatory (BCO). Subsection 11.4 describes the Exoplanet catalogue, a catalogue generated for every star in SDSS, again based upon a different set observational parameters of BCO. Samples of these results are given in Appendix D.

Completion of these catalogues provides the following

- Direct guidance for future observations:
Future differential photometry from BCO or observatories with a similar telescope and observational parameters can use the pointings generated in this catalogue directly. These pointings and their scores indicate the best (and worst)

pointings for telescopes of this type, when observing the class of phenomena for which they were created

- Demonstration of a repeatable system

Catalogues can be generated for other observatories or other target sets using the system developed in this project. Section 14.1 discusses a number of ways in which this project can be reused to generate multiple catalogues suitable for use for individual observers or robotic surveys with observational criteria which do not match those of BCO

- Scoring system for refinement

As discussed in Sections 5.3, the choice of scoring system was originally driven by a requirement for computational simplicity and to meet initial criteria from Young. [51] As discussed in Subsection 14.3.2, it is intended to refine the scoring system based upon comparisons between the output of this project, and the results of observations based upon that output.

11.2. Local Catalogue

The Local Catalogue, as shown in the excerpt in Table 11-1 contains only RA, Dec, and mag information, as these are the criteria used in the Locus Algorithm. As the Locus Algorithm and Local Catalogue design is intended for extensibility to other catalogues, it includes only the information likely to be common to most photometric catalogues.

Position		Magnitude				
RA	Dec	<i>u</i>	<i>g</i>	<i>r</i>	<i>i</i>	<i>z</i>
7.1372	13.9543	19.3300	18.0661	17.5408	17.3283	17.2404
7.1568	14.1287	17.6221	15.7981	14.9408	14.6053	14.3804
7.1907	14.0482	17.0511	15.7541	15.2098	15.0463	14.9774
7.1918	14.0477	23.7024	21.4588	19.6197	18.8573	18.7838
7.2171	14.0830	15.5161	14.2971	13.8758	13.7773	13.7294
7.2488	14.0517	18.2661	16.7711	16.0948	15.8493	15.7224
7.2467	14.0517	25.4099	22.8482	22.1397	21.1464	20.9168
7.2496	13.9712	16.6331	15.0791	14.4748	14.3063	14.2374
7.2553	14.1072	17.0961	15.2991	14.5768	14.3693	14.2484
7.2574	14.1049	23.5485	20.7917	19.4447	18.9023	18.5960

Table 11-1: Excerpt from the Local Catalogue. Columns are RA, Dec and Magnitude (*u*, *g*, *r*, *i*, *z*)

SDSS contains a large number of duplicate entries, and entries for objects which are not stars. The Local Catalogue is intended to contain only stars as potential references and as candidates for the exoplanet catalogue.

To remove the data which is not needed, the Local Catalogue was created by applying a reduced version of the SDSS clean sample of point sources algorithm as described in Subsection 7.2.1.2. The Algorithm used to generate the Local Catalogue selects objects based on a set of bit-flags as shown on Table 7-1. As a result, the Local Catalogue is smaller in both number of rows and columns than the SDSS source data.

Table 11-2 shows the results of a SQL query to the CAS for DR7, comparing the SDSS Clean Sample of Stars algorithm and the reduced version used to generate the Local Catalogue. As can be seen the reduced version permits a small number of entries (0.72%) to pass through to the local catalogue that would be excluded by the full algorithm but does not exclude any stars that should be accepted.

Category of entry from Source Catalogue	Count	Percentage
Entries in SDSS Source Catalogue	585,634,220	
Primary (unique) entries in SDSS	357,175,411	60.99%
SDSS Clean Sample of Stars algorithm	131,934,656	22.53%
Local Catalogue algorithm	136,135,985	23.25%
Clean Sample rejects, Local Catalogue includes	4,201,329	0.72%

Table 11-2: Comparison between Source Catalogue, Local Catalogue, and output from the SDSS Clean Sample of Stars Algorithm based on SQL query to the CAS [6]

Generation of the Local Catalogue provided the following outputs

- Catalogue for use
This project made direct use of the local catalogue, both as a list of candidate reference stars for all uses of the Pipeline, and as a target list for the Exoplanet Catalogue.
- Test of API system
The Generation of the Local Catalogue demonstrated the utility of the API system used in this project. The extensible design allows for modular changes to the API to be made, allowing new local catalogues to be developed in future projects.
- Test platform for Grid Operations
As discussed in 13.3.1, generation of the Local Catalogue was carried out as a grid operation and as a result provided test metrics for grid operations, including an assessment of optimal grid job size.

11.3. Quasar Catalogue

The Quasar Catalogue was intended to provide optimised pointings for all 77,429 quasars in the fourth SDSS quasar catalogue, itself based upon the DR5 SDSS catalogue. [137] The list of quasars provided a target list. The following telescope parameters were used:

Parameter	Value
Field of View	10 arcminutes (0.16666 degrees)
Resolution	1 arcsecond (0.0002777 degrees)
Magnitude variance limit	+/- 2 magnitudes
Colour variance limit	+/- 0.1 magnitudes
SDSS colour band	<i>r</i>

Table 11-3: Parameter list for Quasar Catalogue

The algorithm produced pointings for 23,697 out of the 77,429 quasars and gave null responses to another 16,303, representing targets for which the algorithm could find no satisfactory pointing. The output from the remaining 37,429 was lost due to data storage failure as discussed in Subsection 9.2.2.2. An excerpt from the catalogue is shown in Table 11-4.

Position		Magnitude					Pointing		
RA	Dec	<i>u</i>	<i>g</i>	<i>r</i>	<i>i</i>	<i>z</i>	Av_RA	Av_Dec	Score
256.0224	59.0204	19.350	19.193	19.173	18.920	18.907	255.9825	58.9371	1.2341
256.1418	59.2010	19.373	19.261	19.035	18.964	19.086	256.1441	59.1177	2.3484
255.5668	59.4488	19.537	19.411	19.085	19.053	19.034	255.6814	59.4744	1.8073
255.4656	59.3727	19.073	18.840	18.730	18.716	18.770	255.6291	59.3082	0.7637
255.5836	59.2607	20.266	19.816	19.314	18.778	18.482	0.0000	0.0000	0.0000
255.2419	60.3599	24.579	21.522	19.838	19.696	19.362	0.0000	0.0000	0.0000
255.2369	60.4444	19.766	19.138	18.783	18.554	18.241	255.3836	60.4569	1.8138
255.0467	60.0616	19.747	19.287	18.672	18.072	17.704	255.2137	60.0010	0.9387
256.0256	60.7983	18.560	18.643	18.292	18.201	18.191	255.9964	60.7384	4.1030
255.9825	60.7533	19.601	19.222	18.766	18.616	18.283	255.8417	60.7211	5.2643

Table 11-4: Excerpt from Quasar Catalogue. Columns are Right Ascension, Declination, Magnitude (*u,g,r,i,z*), Pointing RA, Pointing Dec and Score. Highlighted in red are two quasars for which no suitable pointings were possible for the given criteria. Highlighted in green is the quasar with the best score in this small sample, SDSS J170355.79+604511.7

As discussed in Subsection 5.2.6, there are a variety of reasons a target may include no pointing. According to Fan [170], many quasars can be selected from stars because they are outliers from the stellar locus in colour space. Since this algorithm selects reference stars based on proximity in that space, it follows that for many quasars; there would be no suitable reference stars.

Parameter	Number	Percentage of Targets
Targets	77,429	100%
Processed targets	40,000.	51.7%
Pointings	23,697	30.6%
No Pointing Results	16,303	21.1%

Table 11-5: Summary of output from Quasar Catalogue

Descriptive statistics of the variables that contribute significantly to the quasar catalogue are shown in Table 11-6. These statistics show the range of magnitudes SDSS quasars are observed at, with most (~90%) in the range ($18 < r < 20$) The mean values for both g-r and r-i for quasars are significantly lower (bluer) than those for the stars used in the exoplanet catalogue, and these colour indices show a relatively narrow standard distribution when compared with the stars shown in Table 11-10, corroborating Fan's statement. [170]

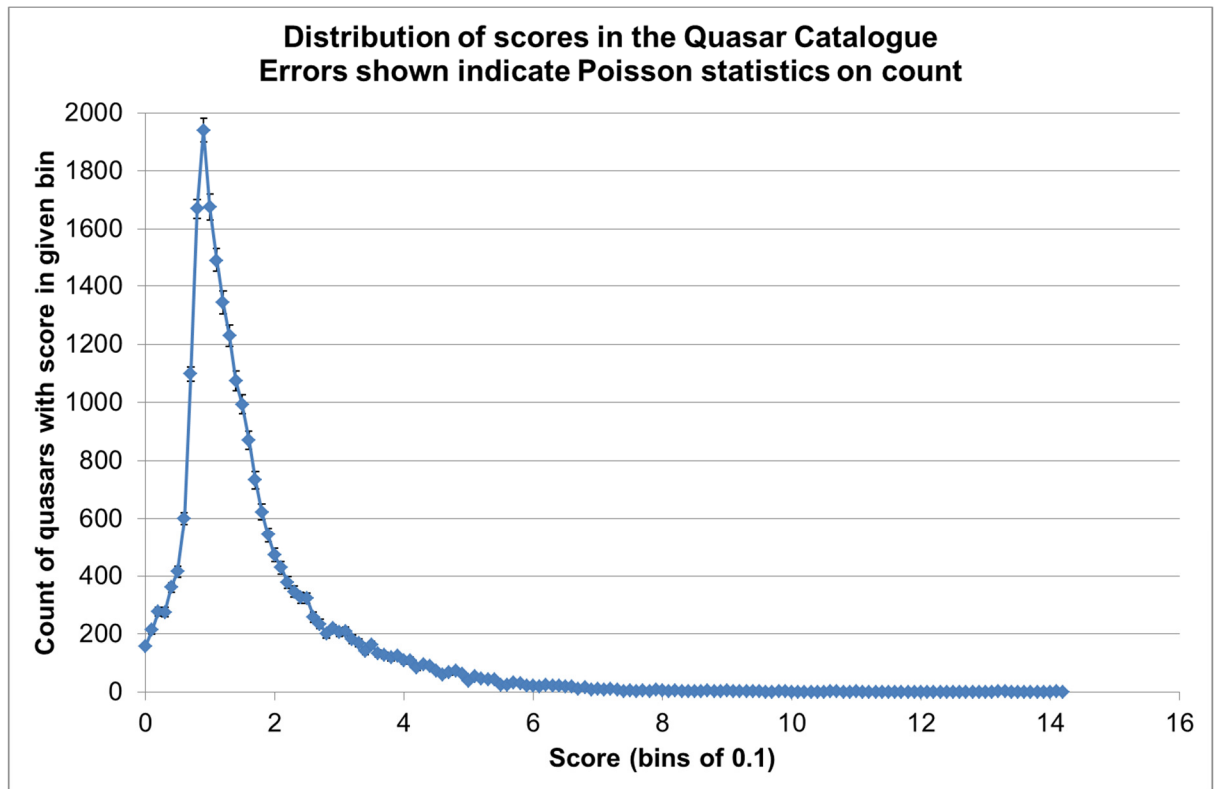


Figure 11-1: Distribution of scores in the Quasar Catalogue. This graph shows only the distribution of scores for quasars for which a valid pointing could be identified.

As a result of the quasars' colour distribution, and because the field of view used for the quasar catalogue is small, (10 arcminutes compared with 15 for the exoplanet catalogue) the scores for the quasar pointings are quite low.

In addition to a large number of quasars with no pointing, the median score for quasars is just 1.33, slightly above that for a star with a *single* “perfect” reference, as discussed in Subsection 5.3.1, with few quasars showing a score greater than 4. The distribution of scores shows with a sharp peak close at a score between 0.9 and 1. As shown in Figure 11-1, scores show a long-tailed, right-tailed distribution skew of +2.0925.

Descriptive Statistic	Magnitude			Colour		Pointing
	g	r	i	g-r	r-i	Score
Maximum	24.4310	22.3160	21.8350	2.8280	1.6910	14.1387
95th Percentile	20.6130	20.3470	20.1610	0.5981	0.3450	4.2493
Mean	19.4618	19.2263	19.0952	0.2355	0.1311	1.7070
Median	19.3920	19.1740	19.0470	0.2040	0.1190	1.3314
5th Percentile	18.2449	18.0350	17.9370	-0.0170	-0.0580	0.4759
Minimum	15.4320	15.2440	15.1840	-0.4240	-0.3020	0.0008
St. Dev.	0.7676	0.7202	0.6926	0.2252	0.1390	1.2226

Table 11-6: Descriptive statistics of the Quasar Catalogue, filtered to those quasars for which pointings were available. The only magnitude (*g, r, i*) and colour parameters (*g-r, r-i*) which contribute to the score are shown.

In summary, the quasar catalogue was valuable for the following reasons

- Optimised pointings for quasars
Pointings were generated for 23,697 quasars which met the required criteria within SDSS for observation by BCO. While the score for many quasars is low, the right-tailed distribution shown in Figure 11-1 indicates a number of quasars for which many suitable reference stars are available: The 95th percentile is 4.25, so 1,184 quasars have a score greater than this.
- Demonstrated the Pipeline
The Locus Algorithm pipeline operated successfully in target list mode to generate the quasar catalogue. The techniques applied in the generation of this catalogue can be applied to do this can be applied to other target lists as proposed in Subsection 14.1.5
- Test platform for Grid Operations
As discussed in 13.3.2, the generation of the quasar catalogue permitted testing of the pipeline in operation in a grid environment. These metrics were used to plan the Exoplanet Catalogue generation.

11.4. Exoplanet Catalogue

The creation of the Exoplanet Catalogue is the primary objective of this project. It represents an attempt to analyse all stars in SDSS for their suitability for differential photometry. By maximising the suitability of the target for differential photometry, it becomes easier to observe the minutes-timescale, millimagnitude variability needed to observe Exoplanet transits. The Exoplanet Catalogue was created using the input parameters shown in Table 11-7

Parameter	Value
Field of View	15 arcminutes (0.25 degrees)
Resolution	1 arcsecond (0.0002777 degrees)
Magnitude variance limit	+/- 2 magnitudes
Colour variance limit	+/- 0.1 magnitudes
SDSS colour band	<i>r</i>

Table 11-7: Summary of Input Parameters for Exoplanet Catalogue

For this project, no assumptions are made about the suitability of a given target: all stars are treated equally. Therefore, the target list for this catalogue consists of all stars in SDSS, as defined by the Local Catalogue algorithm, derived from the SDSS Clean Sample of Stars Algorithm as discussed in Chapter 7. [157] Since the Local Catalogue is used to identify reference stars, the Exoplanet Catalogue may be considered as the result of comparing the Local Catalogue with itself. There are ~86,000,000 stars in the local catalogue.

Parameter	Number	Percentage (overall/LC)
Unique targets (CAS) [6]	357,175,411	100%/n/a
Local catalogue	~86,000,000	24%/100%
Stars Analysed	67,043,579	18.8%/78%
Pointings	61,662,376	17.2%/71%
Null results	5,381,203	1.50%/6.3%
Targets in failed grid jobs	~19,000,000	5.32%/22%

Table 11-8: Summary of output from Exoplanet Catalogue. The percentage column compares elements in the output catalogue with the number of unique objects in the overall catalogue and with the number of objects in the local catalogue.

Note that there are fewer entries in the Local Catalogue than would be predicted by the application of the algorithm to the CAS. This is because the Local Catalogue was based on data in the calibrated object list (*tsObj*) files in the DAS. These files are based upon the SDSS Legacy Survey. Application of the same algorithm to DR7 in the CAS

produces 136,135,985 targets as shown in Table 11-2. However, the CAS includes *all* data for SDSS-II, including Legacy, SEGUE and a Supernova survey. [111]

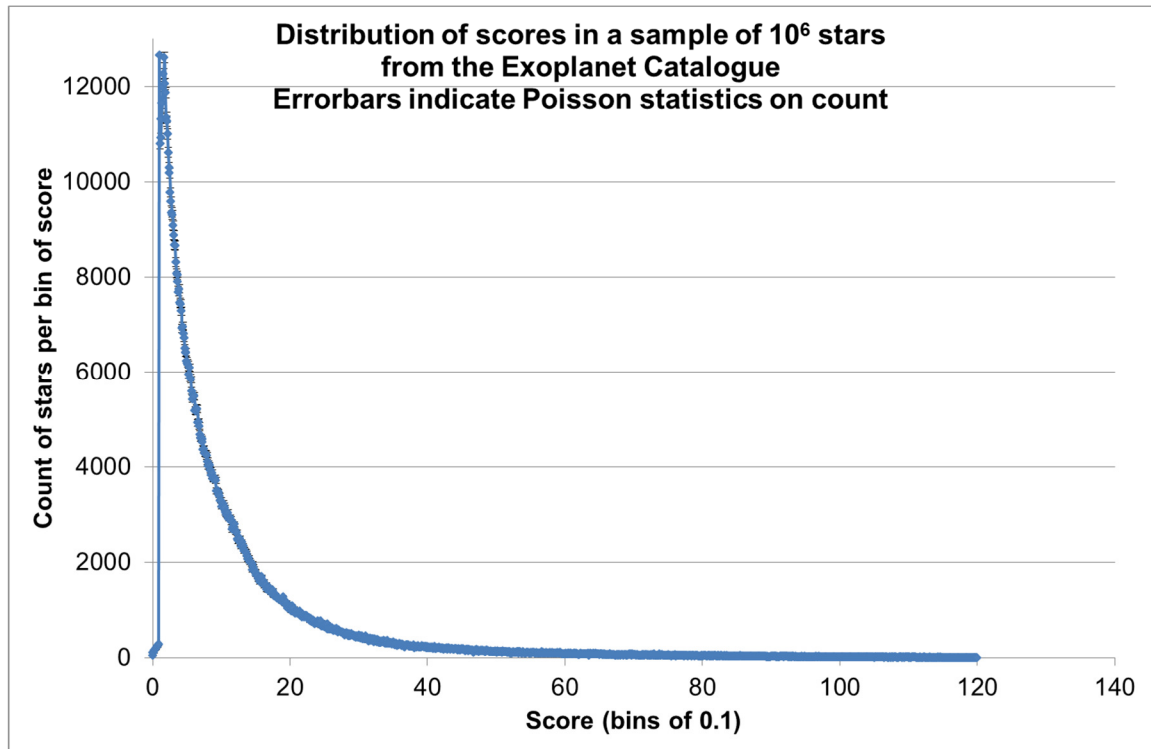


Figure 11-2: Distribution of scores for a sample of 10^6 stars in the Exoplanet Catalogue. Not shown in this sample are 57105 stars for which no pointing was observed

As discussed in Chapter 9, time limits on grid jobs caused ~22% of grid jobs to fail. This left 67,043,579 stars for which an entry in the output catalogue is available, of which, 61,662,376 produced a pointing while 5,381,203 produced a null result: as before representing targets for which there is no viable pointing available. This output is summarised in Table 11-8.

Descriptive statistics of a subset of 10^6 stars in the exoplanet catalogue are shown in Table 11-10. Stars in SDSS show a wider distribution of magnitude than is present in the quasar catalogue. Most stars (~90%) in SDSS fall in the range $16.5 < r < 23.5$. It can be noted that many stars in the catalogue are observed at fainter magnitudes than are shown in the Quasar catalogue. Colour indices $g-r$ and $r-i$ also show wider distributions and higher (redder) means than for quasars.

The larger area of the FoV for the exoplanet catalogue ($15 \times 15 = 225$ square arcminutes as opposed to $10 \times 10 = 100$ square arcminutes) would predict an increase of score by a factor of ~2.25; however the mean (11.41) and median (6.73) scores are much higher

than the corresponding values in the quasar catalogue, even allowing for that correction. In addition, extremely high scores are not uncommon

Position		Magnitude					Pointing		
RA	Dec	<i>u</i>	<i>g</i>	<i>r</i>	<i>i</i>	<i>z</i>	RA	Dec	Score
230.55	-1.59	19.19	17.46	16.73	16.42	16.24	230.50	-1.70	19.43
230.55	-1.59	24.29	21.15	19.80	19.12	18.65	230.42	-1.48	8.63
230.55	-1.74	18.64	16.78	16.00	15.69	15.50	230.43	-1.82	11.48
230.55	-1.55	19.41	18.12	17.52	17.25	17.08	230.64	-1.54	27.66
230.55	-1.67	19.30	18.16	17.67	17.43	17.29	230.68	-1.55	24.52
230.56	-1.74	19.54	18.51	18.01	17.77	17.62	230.68	-1.67	22.05
230.56	-1.63	19.77	18.69	18.21	17.99	17.85	230.68	-1.55	26.06
230.56	-1.63	23.76	20.82	19.31	17.98	17.28	230.63	-1.53	9.29
230.58	-1.60	18.48	17.06	16.47	16.21	16.08	230.57	-1.48	24.82
230.59	-1.65	18.46	17.01	16.38	16.10	15.94	230.52	-1.54	20.87

Table 11-9: Excerpt from Exoplanet Catalogue. Columns are Right Ascension, Declination, Magnitude (*u,g,r,i,z*), Pointing RA, Dec and Score. Highlighted in green is the highest scoring target in this sample

As shown in Figure 11-2, the distribution of scores shows a long-tailed, right-tailed distribution. The distribution shows a broadened, irregular peak between 1.0 and 2.5, after which the score trends downwards. The right-tail of this distribution is much more prominent than that of the quasar catalogue, with a skew of +2.8691. This again indicates that far more high scores are observed among stars than among quasars.

Descriptive Statistic	Magnitude			Colour		Pointing
	<i>g</i>	<i>r</i>	<i>i</i>	<i>g-r</i>	<i>r-i</i>	score
Maximum	27.1639	27.6275	26.3621	8.4090	5.8135	117.7195
95 th Percentile	24.8162	23.5021	22.5144	1.8639	1.6366	38.6017
Mean	21.8563	20.8017	20.1017	1.0546	0.7000	11.4125
Median	22.4006	21.2793	20.4899	1.1180	0.5946	6.7279
5 th Percentile	17.2291	16.5447	16.2241	0.2750	0.0536	1.3971
Minimum	13.2870	12.7994	12.5214	-4.3389	-7.6566	0.0022
St. Dev.	2.3095	2.1428	1.9744	0.5406	0.5514	13.6401

Table 11-10: Descriptive statistics of a subset of 10⁶ stars of the Exoplanet Catalogue, filtered to include only those 942,895 stars for which pointings were available. Only the magnitude (*g, r, i*) and colour parameters (*g-r, r-i*) which contribute to the score are shown.

Further analysis of the colour and magnitude distributions of the Exoplanet catalogue, and the relationship between these distributions and score form the basis of Chapter 12

The Exoplanet catalogue provides the following

- Optimised pointings for stars

Pointings were generated for each of for 61,662,376 stars, each with a score representing how each meets a set of criteria intended to reflect their suitability for differential photometry from BCO.

- Demonstration of the pipeline

The generation of the Exoplanet Catalogue showed the pipeline software working in catalogue mode. This mode, as discussed in Subsection 7.2.2 is used when an entire catalogue is used as a target list. This is a more complex process requiring grid-scale parameterisation and management of the grid load

- Scalability testing

This catalogue provided a test of the scalability of the program to larger datasets in a grid context. Notably, as discussed in Subsection 13.3.3, it exposed some of the limits of the grid system, and a number of changes were made to enable completion of the catalogue.

11.5. Summary

The Quasar Catalogue consists of 23,697 quasar targets with their respective pointings and a score indicating how well the 10 arcminute square field of view around that pointing matches the quasar, with reference stars within ± 2.0 magnitudes in r of the quasar and colour indices $g-r$ and $r-i$ within ± 0.1 .

The Exoplanet Catalogue consists of 61,662,376 targets from the Local Catalogue with their respective pointings and a score indicating how well the 15 arcminute field around that pointing matches the star, with reference stars within ± 2.0 magnitudes in r of the star and colour indices $g-r$ and $r-i$ within ± 0.1 .

The Local Catalogue consists of $\sim 86,000,000$ stars which represent $\sim 22\%$ of the SDSS Source catalogue. These targets were selected using a simplified version of the SDSS clean sample of stars algorithm, with which it agrees over 99% of the time.

12. Meta-Analysis of the Exoplanet Catalogue

The catalogues produced in the course of this project consist of large lists of targets, pointings, and scores for those pointings. Further analysis of this analysed data is needed to provide context for the use of the output catalogues. This second analysis is referred to here as “meta-analysis.” Any future catalogues should be subject to similar meta-analysis to characterise their behaviour as discussed in Subsection 14.1.5

The Locus Algorithm used to produce the output catalogues generated in this project used position and magnitude information on targets and candidate reference stars to generate optimised pointings. Calculated colour indices, generated by calculating the difference between two neighbouring magnitudes (e.g. $g-r$, $r-i$ in *ugriz*) were used to calculate the scores for those pointings.

The meta-analysis looks at the distribution of results in the Exoplanet catalogue when plotted against magnitude and colour. The results of this meta-analysis are used to guide suggested refinements to the project as discussed in Section 14.3.

This Chapter is broken into two Sections, Section 12.1 dealing with variations against Magnitude and Section 12.2 with variations against Colour. SDSS, and thus the Local and Output Catalogues, is not an all-sky survey, instead focussing on an area in the North Galactic Cap. [111] As a result of this selection bias, the meta-analysis does not consider the distribution of results by position.

Both Sections follow the same structure as shown below

- Analysis of source data
Distributions of magnitude and colour for the targets in SDSS are plotted to demonstrate the limits of the source data, and indicate the regime in which good results might be expected.
- Overall patterns of output
By plotting the variation against magnitude and colour of score and the proportion of stars for which no pointing could be identified (referred to as “failed targets”), it is possible to identify regions of the colour and magnitude space which are most likely to provide optimum photometry
- Variations in descriptive statistics

Corroboration of the visual plotting is provided by calculating descriptive statistics: Mean, Median, 5th and 95th percentile for each range of magnitude and colour values. By plotting the variation of these descriptive statistics against the source parameter they pertain to, it is possible to identify regions in which to focus subsequent refinements to the scoring system as discussed in Subsection 14.3.2.

The observation of trends in data demands repeated examination of data from multiple perspectives. Therefore meta-analysis on all 67,043,579 entries in the output catalogue is computationally intensive and, as a result, highly time consuming. Therefore the approach of pragmatic computing demanded that the meta-analyses be carried out on subsets of the data.

Two main sizes of subset of the output were used in this project. First, a subset of 64,000 entries from an arbitrarily selected file in the output catalogue was used to plot the data on a variety of parameters for exploratory analysis. Where a trend was observed in this data, a larger subset of 1,000,000 entries selected from 17 arbitrarily selected files from the output catalogue was used to provide improved precision and allow more accurate results. Unless otherwise specified, all graphs and tables shown below are based on this larger subset.

Due to the continuous nature of the data, trends in the data were identified by plotting graphs of the count of stars in a given bin on one or more parameters. Unless otherwise specified, all bins are defined by the following inequality: $b \leq x < b + s$, where b is the bin label, x is the value of the parameter for the given target, and s is the bin size. (e.g. a bin of 0.1 magnitude labelled 17.6 would include all stars with magnitude greater than or equal to 17.6 and less than 17.7) While Poisson errors are applied for these counts, in many cases the data volume (1,000,000 records) ensures that the errors are too small to display. This is noted in the captions on the plots.

12.1. Magnitude Variations

This Section examines the distribution of magnitudes in SDSS, and the relationship between this distribution and that of scores and failed targets in SDSS. As the Exoplanet catalogue was generated using SDSS r magnitude, this analysis focusses primarily on that band

In any given photometric system, more stars are predicted to be observed at fainter magnitudes than at brighter ones. Equation 12-1 shows a least-squares-fit equation for the distribution of stars for the whole sky in Johnson V magnitude according to Allen [171].

$$N = 10^{0.754 + 0.4896 V + 0.001159 V^2 - 0.000235 V^3}$$

Equation 12-1: Prediction of the distribution of stars by Johnson V magnitude across the whole sky according to Allen. [171]

Figure 12-1 shows a plot for SDSS showing the distribution magnitudes of stars across all five sdss filters (u , g , r , i , and z), together with Allen's prediction. Each observed SDSS plot shows near-exponential growth over a characteristic range of magnitudes. Unlike the prediction from Allen, there is also a distinctive sharp fall-off at high magnitudes. The faint-magnitude drops are indicative of the limiting magnitude of the SDSS telescope at Apache Point under the SDSS observing regime (the limit at which SDSS achieves 95% detection of point sources is 22.2 magnitude in r , for example.) [172]

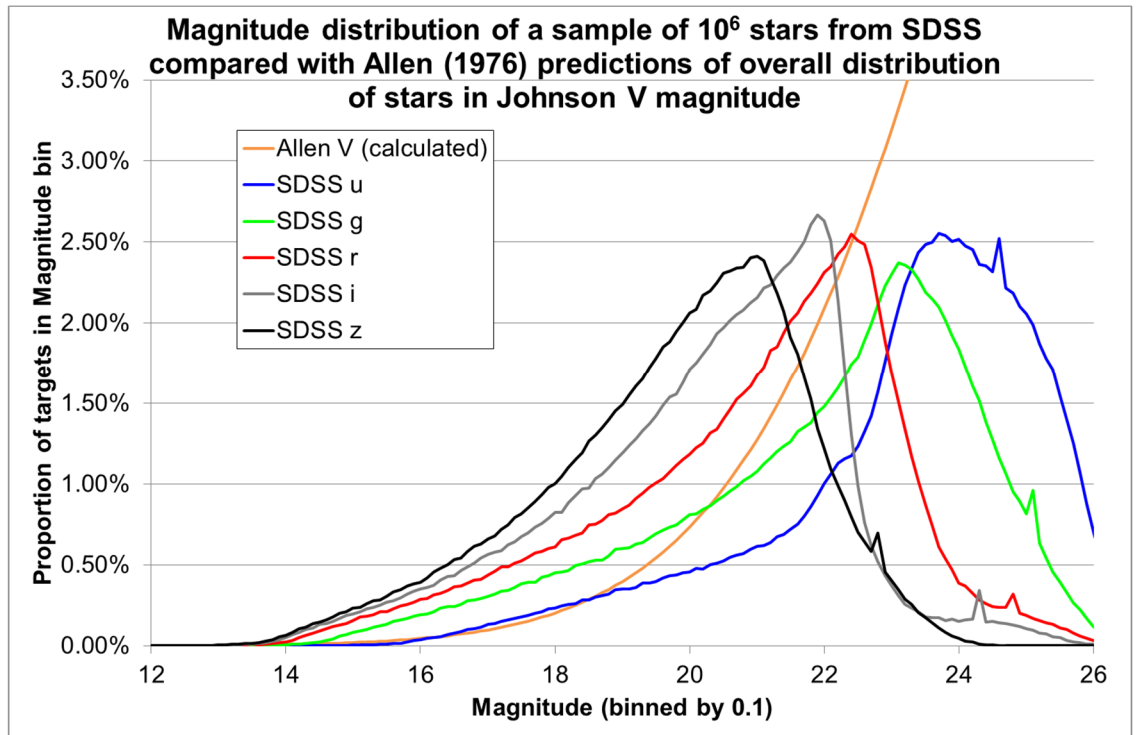


Figure 12-1: Distribution of Magnitudes for a sample of 10^6 stars from SDSS. The proportion of the sample is plotted against the proportion of the distribution of stars in the whole sky as predicted by Allen. [171] Errorbars are not included as the Poisson error is too small to be displayed.

At low magnitudes, the count of stars in the Local catalogue declines to 0. This drop-off for bright stars is caused primarily by the saturation point of the SDSS catalogue. The majority of SDSS entries brighter than 14th magnitude in r are flagged as saturated, and SDSS photometry of saturated objects is questionable. [157] [173] Therefore objects flagged as saturated are removed by the algorithm to produce the local catalogue.

These two limits imply limitations on the utility of any results that depend on photometry of stars outside the range $14 < r < 23$.

12.1.1. Variation of Score with Magnitude

Figure 11-2 shows the overall distribution of scores for the exoplanet catalogue as a blunted peak from 1.0-2.5, followed by a steady decline as score increases. Exploratory visual analysis demonstrated that this distribution was not constant across all magnitudes. Figure 12-2 shows the distribution of scores as a proportion of the overall number of targets in a set of r magnitude bins.

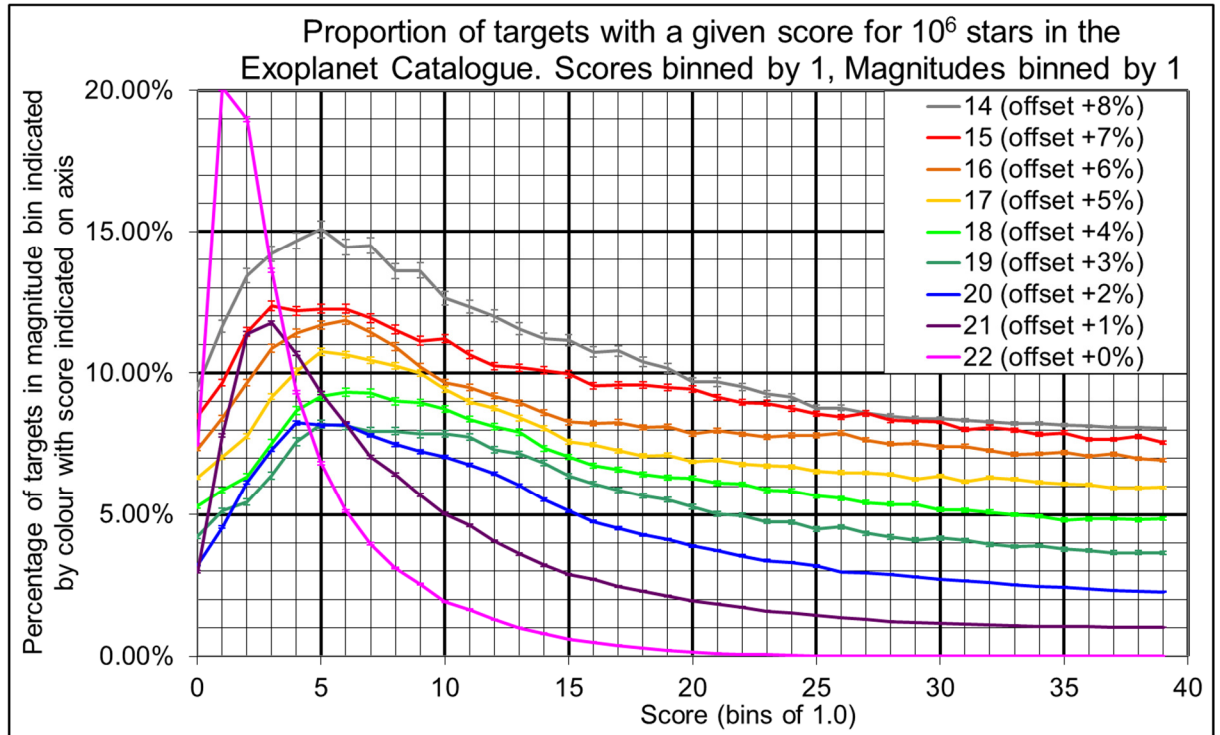


Figure 12-2: Distribution of scores as a proportion of overall targets in a given r magnitude bin. The addition of a variable offset allows for separation of the magnitude binning. Errors shown are given by Poisson statistics

Sets of stars of different magnitudes show different distributions of score in SDSS. The distributions of scores in the magnitude range $14 < r < 20$ appears similar, and each shows a peak at a score of 5-6. Fainter objects show a peak score close to 1. As fainter objects are more common, as shown in Figure 12-1, they dominate the overall distribution curve shown in Figure 11-2.

These distributions indicate a higher peak score, and more stars with higher scores in the region $14 < r < 19$, with the highest peak at $r \approx 18$. This suggests that a user seeking high scoring targets using these parameters should focus on this part of the magnitude space. The highest score for any single star was 117.7 for a star with $r=17.70$

12.1.2. Variation of Number of Failed Targets with Magnitude

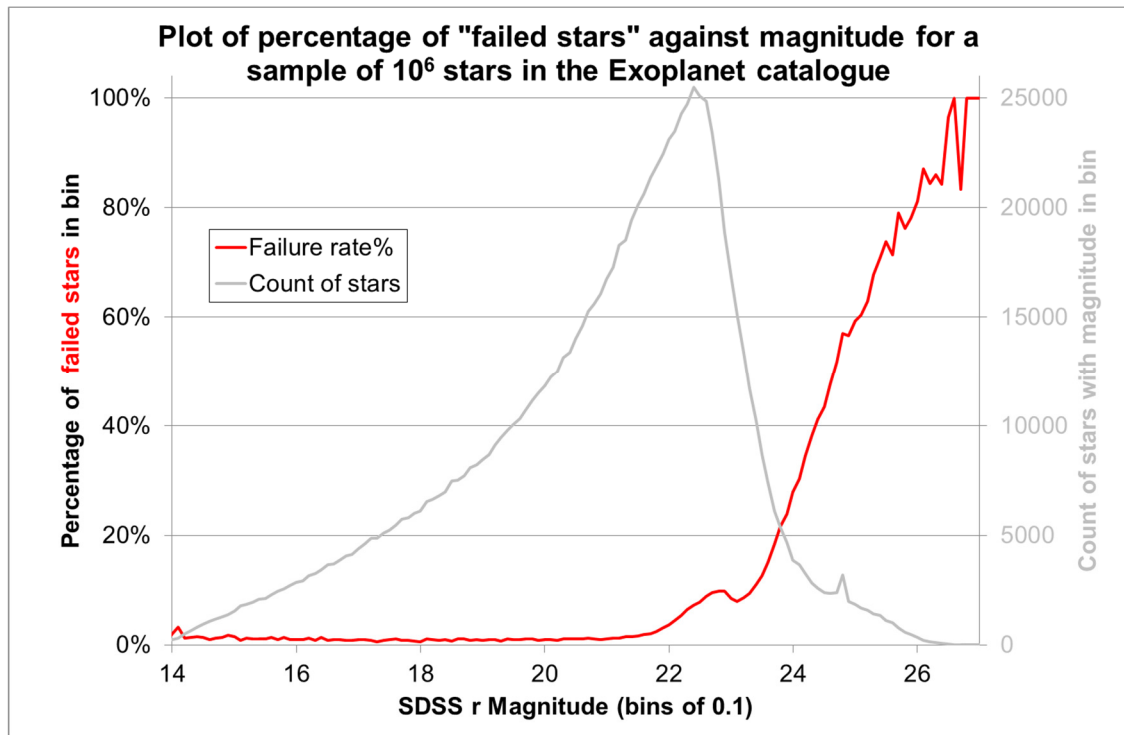


Figure 12-3: Plot of failed targets against magnitude. Shown in grey in the background is the overall distribution of stars by r magnitude. Errorbars are not included as the Poisson error is too small to be displayed.

A score of exactly zero indicates a target for which no viable reference could be found, known as a failed target as shown in Figure 5-9. Any target for which there are any suitable reference stars for which a pointing can be generated will have a score greater than zero. Therefore two different trends can be observed by observing trends in the

score variable. The proportion of scores of exactly zero indicates the number of failed targets in a given bin magnitude.

A plot of failed targets as shown in Figure 12-3 against magnitude demonstrates a low, fairly consistent, failure rate ($\sim 1.5\%$) in the range $14 < r < 21$ which rapidly increases above that magnitude, and becomes dominant above the SDSS limiting (95%) magnitude of 22.2. Brighter than 14th magnitude, most objects are saturated, and few are retained in the Local and Exoplanet catalogues, rendering statistical analysis of failed targets impractical.

12.1.3. Variation of Descriptive Statistics

As shown in Figure 12-2, the distribution of scores varies considerably with magnitude. Figure 12-4 shows a suite of descriptive statistics of these distributions for a sample of 10^6 stars from the exoplanet catalogue.

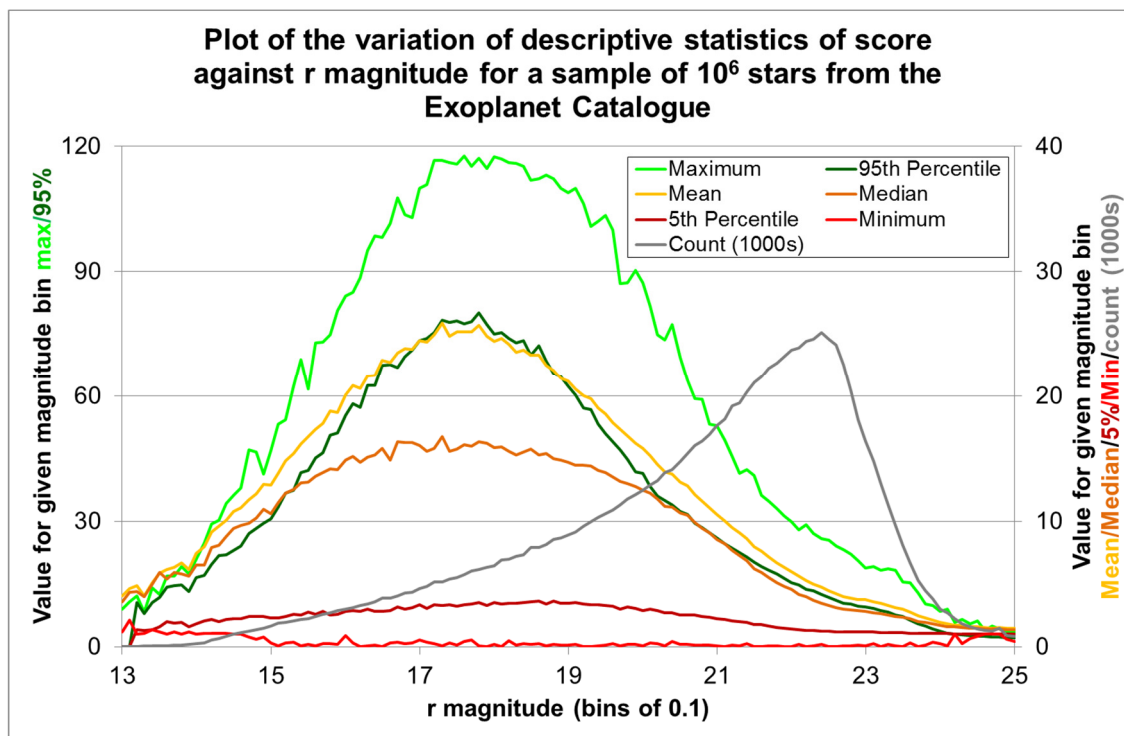


Figure 12-4: Distribution a set of descriptive statistics of scores for a sample of 10^6 stars in the Exoplanet Catalogue, separated into magnitude bins and plotted against magnitude. Note that Maximum and 95th percentile values are plotted on the left-hand y-axis, while other variables are plotted on the right axis as the former have a much wider range of values.

Most of these measures show a broad peak at $r \approx 17.5$, indicating the region of the magnitude space in which the targets best match the observational criteria provided. For the most part, the distributions fall off rapidly outside the range $16 < r < 20$.

The significance of each of these descriptive statistics is as follows

- **Maximum**
This shows the single highest score for any star in the r magnitude range. These can be individual outliers, but also indicate the best possible match to observational criteria.
- **95th Percentile**
The top 5% of scores in a magnitude bin. Can be used to highlight regions with significant numbers of very high scores, which may be suitable for use with automated surveys.
- **Mean**
Arithmetic mean of scores in the r magnitude bin. This parameter can be used to determine if a particular star is “above-” or “below-average” when compared with stars of its magnitude. Note that mean is strongly affected by skewed, long-tailed distributions such as those shown in Figure 12-2
- **Median**
The central value of score in the r magnitude bin. Can also be used to determine if a particular star is “above-” or “below-average” when compared with stars of its magnitude. Median is not affected by outliers and other extreme values.
- **5th Percentile**
Indicates the bottom 5% of scores in a magnitude bin. Used to demonstrate the skew of the distributions, as the difference between the median and 5th percentile is significantly smaller than that between the 95th percentile and the median.
- **Minimum**
Indicates the lowest score found which is not a “Failed target.” No significant patterns are visible in this variable, but it is included for completeness.
- **Count**

This shows the distribution of stars by r magnitude in thousands of all stars in the sample. Notably, the distribution of stars in r magnitude is not strongly correlated with the distribution of the descriptive statistics of those variables

12.2. Colour Variation

As shown in Section 5.3, rating for a candidate reference star is calculated based on the similarity of the colour index of the candidate reference star to that of the target for the colour bands either side of the colour band for which the catalogue is generated. Score for a pointing is calculated by adding together the ratings for all candidate reference stars which can be included in a field of view centred on that pointing.

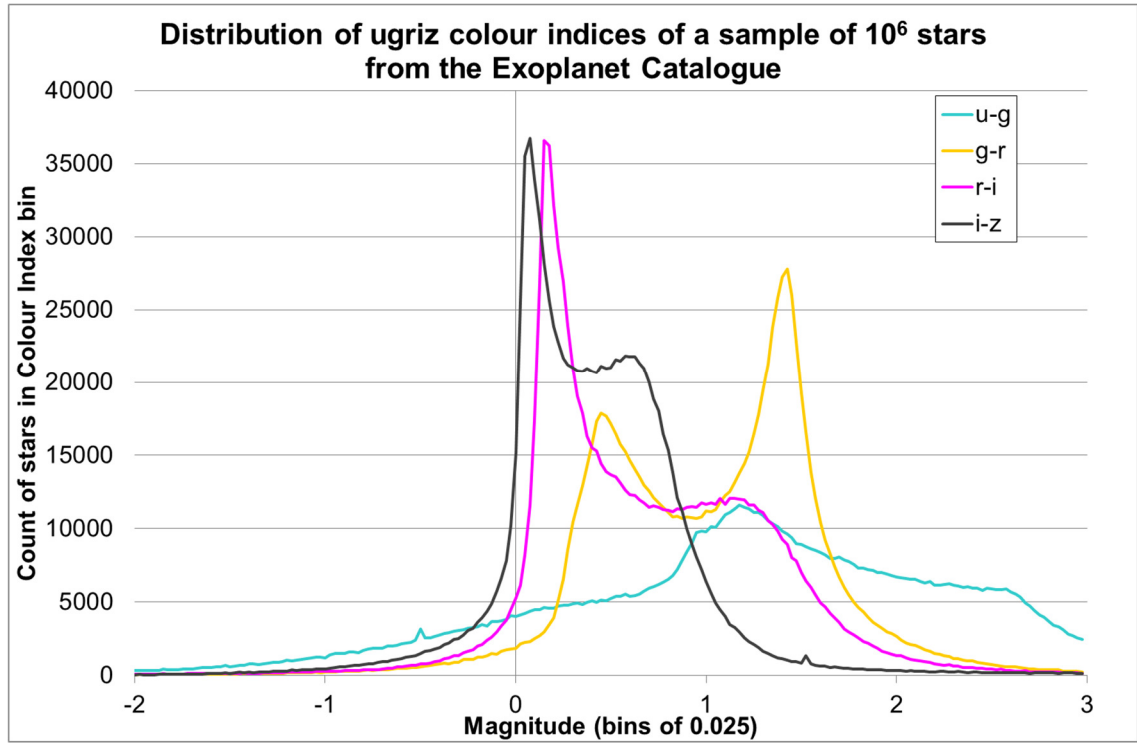


Figure 12-5: Colour Distribution of Stars in SDSS. All four colours using neighbouring magnitudes are shown here. No errorbars are shown, as Poisson statistical variation would not be visible at this scale.

Since the Exoplanet Catalogue was developed for the SDSS r band, the $g-r$ and $r-i$ colour indices are used in the calculation of score in the Locus Algorithm. The meta-analyses below demonstrate the variation in the distribution of score and failed targets against the distribution of stars with these colour indices.

According to Fan, the colour distribution of stars in SDSS is not predicted to be flat, but instead shows a bimodal pattern, with two peaks. [170] Fan suggests that stellar metallicity is a major contributor to this variation, with stars with higher metal contents being bluer, but detailed analysis of this pattern is not within the scope of this project. Since there are more stars observed with colours close to the peaks of these distributions, stars in these regions of the colour space are more likely to have neighbours of similar colour, and therefore higher scores are predicted for stars with colour close to these peaks.

Figure 12-5 shows the distribution of a sample of 10^6 stars from the Exoplanet Catalogue by colour indices in the four neighbouring-colour indices in SDSS, counting the number of stars in bins of 0.025 in colour index.

- $u-g$ shows a broad, low distribution with a peak at $u-g=1.175$, with multiple unresolved peaks in the distribution.
- $g-r$ shows a sharp bimodal distribution. The primary peak is observed at $g-r=1.425$, while a secondary peak is visible at $g-r=0.45$, with a local minimum between these two values at $g-r=0.90$
- $r-i$ shows a sharp primary peak at $r-i=0.15$, and a much lower, partially resolved peak at $r-i=1.125\pm0.075$
- $i-z$ shows a narrower overall distribution than the other indices. This distribution is also bimodal, with a primary peak at $i-z=0.075$ and a secondary peak at $i-z=0.575$

12.2.1. Variation of Score with Colour

As shown in detail in Section 10.3, in the Exoplanet Catalogue, score for a pointing is calculated based on the $g-r$ and $r-i$ colour indices of the target and each candidate reference star. As with the variation in the distribution of score with magnitude shown in Subsection 12.1.1, the shape of the distribution of score varies significantly with colour index. This allows a user of the catalogue to identify regions of the colour space in which particularly high scores may be found

Figure 12-6 shows the complex distribution of the number of stars with a particular combination of score and colour index. By comparing the distributions of scores in

each colour bin to the overall distribution of score shown in Figure 11-2 and the distribution of colours shown in Figure 12-5 it is possible to observe some patterns.

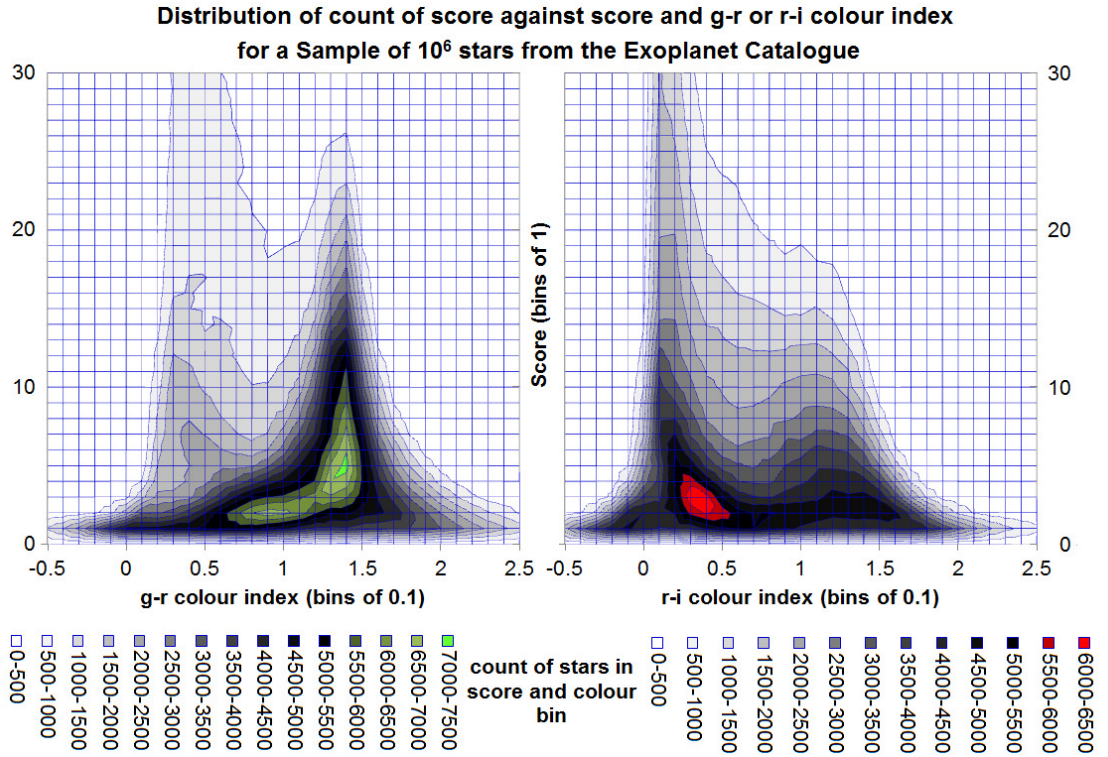


Figure 12-6: Distribution of count of scores against score and *g-r* (left) and *r-i* (right) colour indices for a sample of 10^6 stars in the Exoplanet Catalogue. Stars for which no pointing could be determined are excluded from these plots. The distribution of scores against colour index can be observed to be similar to the distribution of stars by colour index as shown in Figure 12-5. The peak of each distribution is highlighted in green and red respectively.

The distribution of scores appears to be broadly similar to the overall distribution of colour for both of the indices. Most stars ($\sim 90\%$) in SDSS are in the region $0.27 < g-r < 1.86$ and $0.05 < r-i < 1.63$. Outside these regions, the peak in the distribution of score lies at $score=1$, and the distribution in scores above that value trends down rapidly. Within those regions, the number of targets with high scores becomes far more significant, with the largest distribution near to the peaks of the distribution of stars by colour index in both cases.

In the region $0.3 < g-r < 0.6$, and $0.1 < r-i < 0.2$, corresponding to the bluer peaks each of the distributions of colour, there appear to be two peaks in the distribution of scores, a primary maximum at $score=1$, and a smaller local maximum at $score=7$. The distribution of scores in this region also shows a very long-tailed distribution, including

the highest score for any single target in the sample (117.7) being found in this region, with $g-r = 0.45$ and $r-i = 0.17$. This suggests contributions from at least two separate distributions as suggested by Fan [170]

12.2.2. Variation of Number of Failed Targets with Colour

As discussed in Subsection 5.2.6, no pointing can be found for a target for which there are no reference stars. In the generation of the Exoplanet Catalogue, the colour of a candidate reference star must be within 0.1 magnitudes of the colour of the target in both $g-r$ and $r-i$ magnitudes.

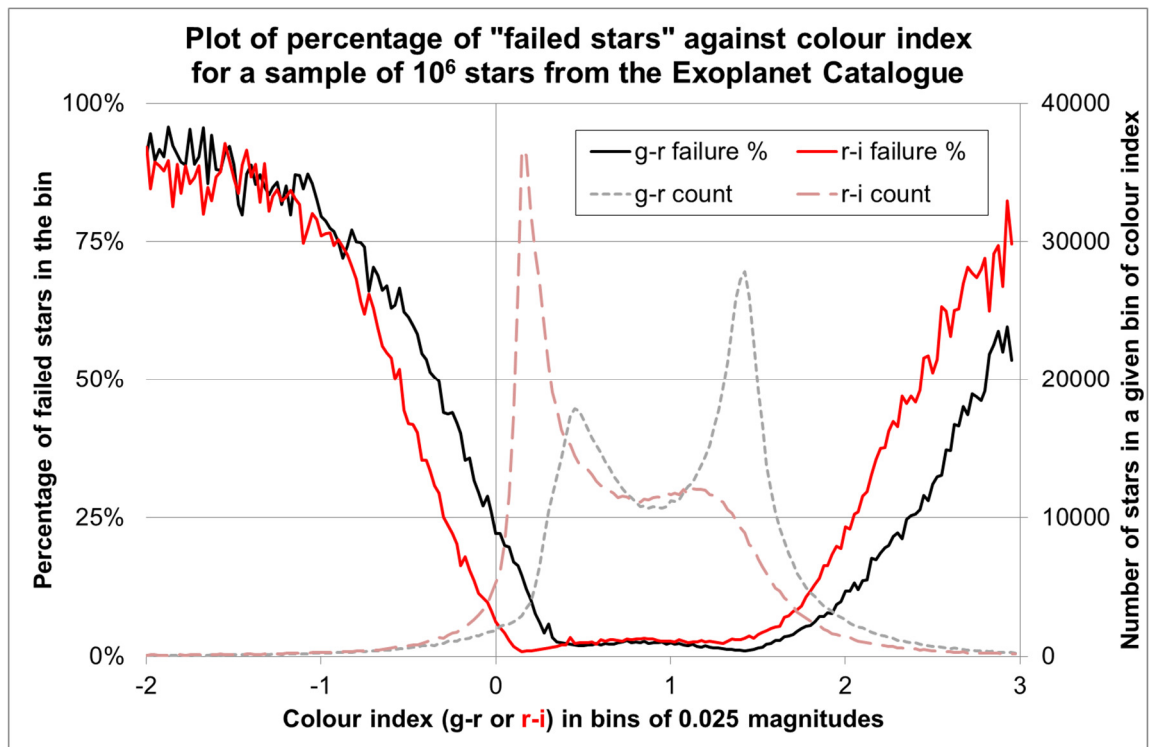


Figure 12-7: Proportion of targets in a given bin of colour index that failed to find a pointing, plotted against the proportion of stars in the sample in that bin.

If the target is in a region of either or both colour spaces which does not have many candidate reference stars, it becomes more likely that a pointing cannot be found. Conversely, if there are many candidate reference stars of a particular colour, it becomes less likely that the Locus Algorithm will fail to generate a pointing.

Figure 12-7 demonstrates the proportion of failed targets by $g-r$ and $r-i$ colour indices. Most stars ($\sim 90\%$) in SDSS are in the region $0.27 < g-r < 1.86$ and $0.05 < r-i < 1.63$. Outside

this region, the proportion of targets for which no pointing could be generated rises rapidly as the number of stars per magnitude bin decreases.

The minimum number of failed targets when plotted against colour occurs, as predicted, at or close to the peak of the overall distribution of stars with 1.01% of stars failing at $r-i=1.425$ and 0.87% of stars failing at $r-i=0.150$. A local minimum of 1.87% of failed stars appears close to the secondary peak for $g-r$ at 0.500. A broad local maximum occurs between these minima in $g-r$. No clear secondary minimum is observed in failure rate against $r-i$.

12.2.3. Variation of Descriptive Statistics with Colour

A. Figure 12-8 demonstrates the dependence of a variety of descriptive statistics of the distribution of score when plotted for various $g-r$ colour index bins, while Figure 12-9 demonstrates the same variations against $r-i$ colour index. The purpose of plotting the variables is identical to that in Subsection 12.1.3 where the distribution of score was plotted against the distribution of magnitude.

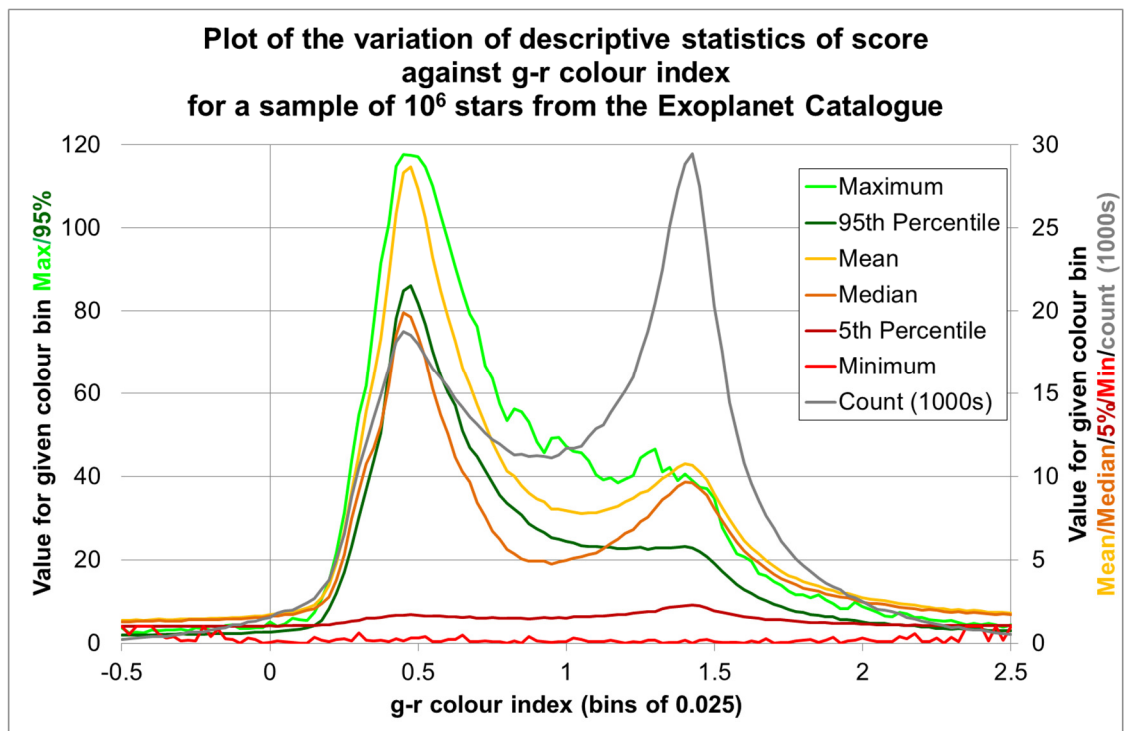


Figure 12-8: Distribution of a variety of descriptive statistics of scores against $g-r$ colour index. Note that Maximum and 95th percentile values are plotted on the left-hand y-axis, while other variables are plotted on the right axis as the former have a much higher range of values.

In all cases, a stronger dependence of distribution of score on the distribution of stars in the colour band is shown than was demonstrated between those measures and magnitude, but the relationship appears to be closer for $r-i$ than $g-r$.

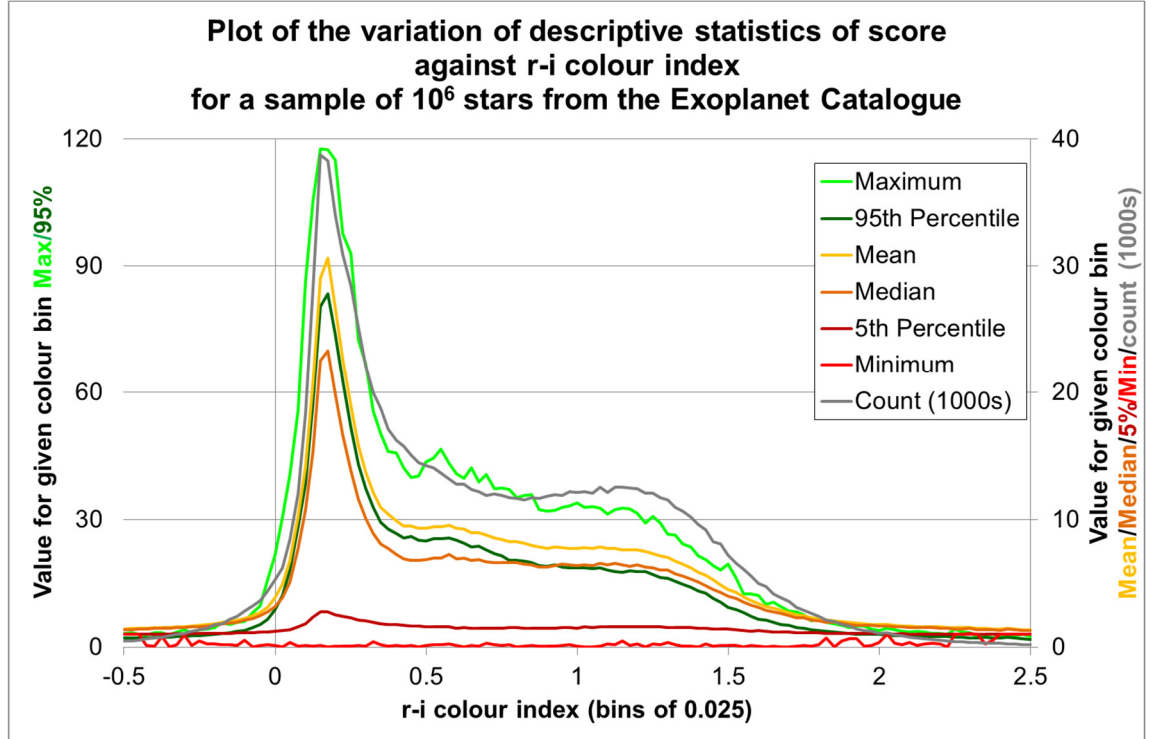


Figure 12-9: Distribution of a variety of descriptive statistic of scores against $r-i$ colour index. Note that Maximum and 95th percentile values are plotted on the left-hand y-axis, while other variables are plotted on the right axis as the former have a much higher range of values.

Notably, for $g-r$, the maximum values for maximum, mean, median and 95th percentile of score distribution all occur close to the secondary (bluer) peak of the distribution at 0.450. This is associated with the long-tailed distribution of scores shown in Figure 12-6, which indicates a large number of stars with very high scores.

All of the descriptive statistics of score drop to near-zero outside the ranges of colour for most SDSS stars i.e. $0.27 < g-r < 1.86$ and $0.05 < r-i < 1.63$, and as stars with colour within $\pm 0.1\text{mag}$ of the target are considered candidate reference stars, scores begin to drop rapidly as they approach those boundaries as there are fewer available references.

12.3. Summary

Three variables pertaining to stars in the Local Catalogue are used in the generation of the Exoplanet Catalogue – Position, Magnitude and Colour. Of these, position is not

evenly distributed across the sky, but instead focussed on the Northern Galactic Cap. [111] As a result, distribution of score against position was not investigated. The parameters of the Exoplanet grid job specified that it was generating a catalogue for use with SDSS r band filters. Therefore variation of score against r magnitude and $g-r$ and $r-i$ colour indices were investigated to identify regions of the SDSS magnitude and colour space in which the best scores can be found.

At magnitudes of $r < 14$, most SDSS stars are marked with the `saturated` flag and as a result are discarded from the Local Catalogue, and are not used as targets or candidate reference stars in the Exoplanet Catalogue. At magnitudes above the SDSS 95% limiting magnitude for detection of point sources at $r=22.2$, the number of stars observed drops rapidly. SDSS Source data is therefore considered reliable in the range $14 < r < 22.2$. Any target outside this range may be considered unreliable and should not be used.

For the Exoplanet catalogue, which used $r \pm 2$ as the acceptable range for potential reference stars, any target outside the range $16 < r < 20.2$ would be expected to suffer loss of reference stars and thus reduced *score*. Use of targets outside this range is therefore discouraged.

The distribution of score shows the highest values in the region $17 < r < 18$, but useable scores are observed across the full range as long as one avoids the limits of SDSS. Job failure rate remains low in the range $14 < r < 21$, but becomes dominant at higher magnitudes.

The colour distribution of the stars in SDSS shows that a majority (90% - 5th to 95th percentile) of these stars have colour indices in the ranges $0.27 < g-r < 1.86$ and $0.05 < r-i < 1.63$. Significantly, the median colours for the quasars used in this project are $g-r=0.2040$ (outside the 5th percentile) and $r-i=0.1190$ (close to the 5th percentile) which explains the reduced number of reference stars available, and thus the low scores and high failure rates achieved in the Quasar Catalogue.

$g-r$ shows a sharply bimodal distribution showing peaks at 0.450 and 1.425, the latter peak being higher. $r-i$ shows a strong primary peak at $r-i=0.15$, and a lower, partially resolved peak at $r-i=1.125 \pm 0.075$. As score in the Exoplanet catalogue is calculated

based on these parameters, it was expected that stars close to these peaks would show high scores and low failure rates, while stars away from these peaks, and especially outside the 90% range would show low scores and high failure rates.

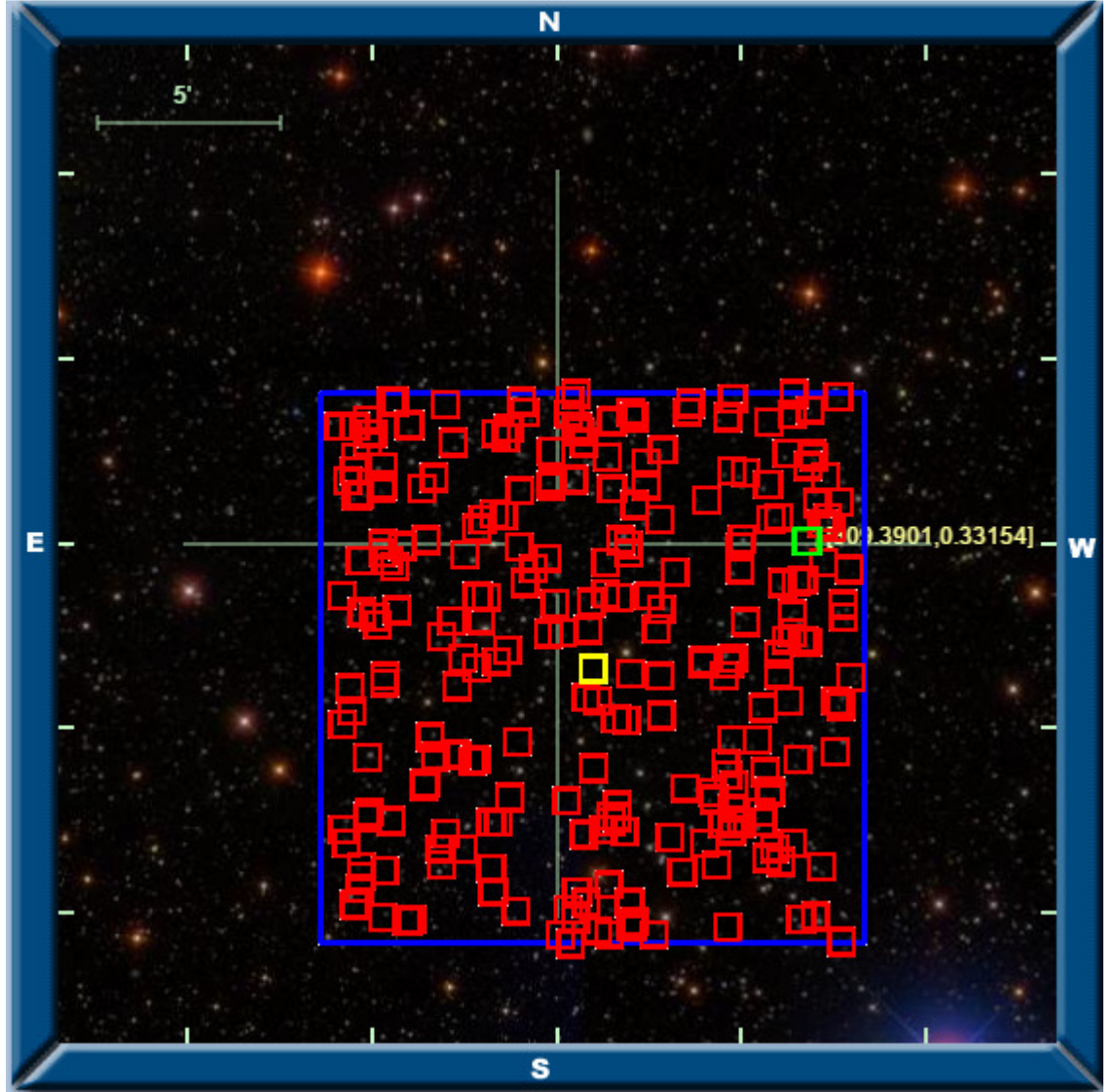


Figure 12-10: SDSS Navigate image for J203733.62+001953.5, the target with the highest score of any target in the sample of 10^6 stars from the Exoplanet Catalogue. Shown in green is the target, in red are each of the 247 reference stars selected for use with that target to produce a score of 117.70, in yellow is the pointing – the point at which the telescope should be aimed and in blue are the boundaries of a 0.25 degree field of view centred on that pointing. This image may be contrasted with Figure 10-6 to illustrate the increase in the number of reference stars for this “top” target.

The distribution of scores appears to correlate with the distribution of both $g-r$ and $r-i$ colour of targets in the catalogue, however the distribution of score is higher for the

smaller peak in the distribution of $g-r = 0.450$, as shown in Figure 12-8. Low failure rates are observed across the range of values of colour for most SDSS targets (i.e. $0.27 < g-r < 1.86$ and $0.05 < r-i < 1.63$) High failure rates and low scores are observed for targets outside these ranges. As a result, use of targets outside those ranges is not recommended.

Overall, the mean and median of score for targets in the Exoplanet Catalogue are 11.4125 and 6.7279 respectively. Targets could therefore be considered “above average” if their score exceeds one or both of these values. However, target score shows a strong dependence on colour and a weaker dependence on magnitude. Ideally, targets should be selected with magnitude close to $r \approx 17.5$ and colour indices close to the blue peaks of both distributions ($g-r = 0.450$, $r-i = 0.150$), as the highest scores are observed close to those values.

The individual star in the subset selected with the highest score given the parameters SDSS Colour = r , FoV = 0.25° , resolution = 0.0002777° , $|\Delta m_{\max}| = 2.0$ and $|\Delta c_{\max}| = 0.1$ was SDSS J203733.62+001953.5, with a score of 117.7 from 247 reference stars, $r = 17.70$, $g-r = 0.45$ and $r-i = 0.17$. The FoV generated for this target is shown in Figure 12-10.

As discussed in Subsection 2.2.3.5 and Section 5.3, this score indicates a large number of available reference stars within a narrow magnitude and colour range from the target. This enables the reduction of many of the sources of error in the measurement of the differential magnitude of this target. [69] [22] Exact analysis of the degree to which precision of these measurements can be improved is the subject of future work as discussed in Subsection 14.3.2. In addition, the 247 reference stars may all be intercompared, providing 247 separate lightcurves from which an exoplanet transit may be observed.

13. Computational Results

This Chapter presents the results of this project from a computational point of view. The objectives of this project from a computing perspective, as stated in Section 4.2, were as follows:

- Develop a system capable of meeting the Astrophysical objective of analysing SDSS to identify optimum pointings for Differential Photometry
- Evaluate that system's performance in unit testing
- Use that evaluation to plan and implement a grid solution
- Assess the performance of the grid solution when compared with predictions from unit testing to identify grid-specific issues
- Produce large catalogues of output data and characterise that data
- Provide guidance for future development.

This Chapter examines the results of this approach from the perspective of three sets of metrics:

- Data metrics, shown in Section 13.1, which consider the size of the data at various stages of the project, in terms of entries in the various catalogues, the number of files used to store that data, the number of directories etc.
- Processing metrics, as described in Section 13.2, which are based on the time taken to download, process and analyse the data in unit testing.
- Grid metrics, as discussed in Section 13.3, which consider how well the conversion from single processor to grid computing scaled up.

Finally, Section 13.4 provides an appraisal of issues arising within the project which impacted its performance. This includes bottlenecks in grid jobs and observed reliability issues with both processing and data. A number of mitigations for these issues are suggested which are expected to be valuable as a guide for future work as discussed in Section 14.1.

13.1. Data Metrics

As discussed in Chapter 8, this Thesis requires the download, processing and access of large amounts of data, which must be carefully organised such that it can be accessed at various stages of the project.

Table 13-1 shows a number of properties of the data at various stages in this project. Each catalogue consists of many FITS data format files. As discussed in Subsection 8.3.1, each FITS file includes one or more HDUs, each consisting of a header that describes the data and a data unit that contains said data. In the case of the fits files used in this project, the data units are all fits tables, consisting of a row for each entry in the catalogue and a column for each piece of data about that column. For example, the Local Catalogue FITS files, as described in Subsection 8.3.1.2 has a double precision floating point column for RA, a double precision floating point column for Dec and an array of five double precision floating point columns for *ugriz* magnitudes.

	SDSS DR7 Catalogue [111]	Local Catalogue	Quasar Catalogue	Exoplanet Catalogue
Entries	357,175,411 [6]	~86,000,000	40,000.	67,043,579
Files	421,388	358,076	40.	1,598
Directories	3,290.	2,609	1	7
Size on disk	4.76 TB	6.89 GB	3.43 MB	5.02 GB
Mean Size per file	11.8 MB	20.2 kB	87.8 kB	3.22 MB
Mean Size per entry	14.3 kB	~86 B	87.8 B	80. B
Mean Entries per file	847	240	1,000.	41,955

Table 13-1: Data Size for the various databases used and created during this project. Approximate or rounded values are indicated with a tilde (~), while values with significant trailing zeroes are indicated with a decimal point (.). The Mean Size per entry is calculated by dividing the total size of the catalogue by the number of entries and as such includes contributions from header data spread per entry.

The Entries row in Table 13-1 refers to the number of rows in the given catalogue between all the files (second row on Table 13-1) combined. Those files are stored in directories governed by the data organisation structure laid out in Subsection 8.2.5. The number of directories listed on Table 13-1 thus varies, largely depending on whether the catalogue was organised into the SDSS hierarchy comprised of `run`, `rerun`, `camcol` and `field` as defined in Subsection 8.2.5.1 or whether other concerns, such as the aggregation of results in output data files dominated the design requirements. The size of the respective catalogues listed in Table 13-1 was measured experimentally by Grid Ireland and the SCG.

Table 13-1 also lists a number of calculated values. The size per file is the mean size given the size of the catalogue on disk and the number of files of the catalogue. The size per entry is similarly calculated based on the aggregate number of entries in the catalogue. This value is larger than might be expected by naively adding the size of the data types for the columns in the catalogue: for example, the Local Catalogue might be

expected to use 56B per entry ($2+5=7$ double precision numbers, $\times 8$ (`sizeof(double)`) = 56B) but they are in fact 86B per entry. This can be attributed to the fact that for files with few entries, the size of the Header can be a significant contribution to the overall file size. The much larger Exoplanet Catalogue output files as defined in Subsection 8.3.1.3 hold 10 double precision numbers per entry, but because there are many rows in the file, the contribution of the header to the size per file becomes negligible.

13.1.1. Source Data Metrics

The source data for this project is the Sloan Digital Sky Survey Catalogue, Data Release 7. As described in Subsection 8.2.5.1, it was decided to store this data in a structure that mimicked its structure in the SDSS DAS catalogue. This structure permitted the use of SDSS search tools such as the CAS to be used to probe the data. For example, SDSS maintains a correspondence between file path/name and location on the sky, which can be accessed using the CAS SQL database.

A key concern in the early phase of the project was the transfer of data from SDSS to Grid Ireland for processing. While the DAS made the data available online, it was a concern that downloading the data *en bloc* would put a strain on communications links either at source or in TCD.

Instead of placing strain on computing resources, the final decision was made based on the fact that the software development phase of the project was not complete when the data was being accessed. A script was developed by John Ryan at Grid Ireland to download and access the data from the DAS in small batches at off-peak times over the course of six months in 2008. [174]

As shown in Table 13-1, this data totalled 4.76 TB in 421,388 separate files. The full directory tree used to store this data amounted to 3,290 directories, counting subdirectories.

13.1.2. API Data Metrics

The API phase of the project, as described in Subsection 7.2.1, processed the source data to produce the Local Catalogue. For each file in the source catalogue, a

corresponding Local Catalogue file was created in a grid job. Each file in the Local catalogue retained the same path and file name convention, slightly modified as discussed in Subsection 8.2.5.1.

This process eliminated any entries in the catalogue that did not represent a clean sample of point sources (i.e. stars) as discussed in Sections 7.2.1.2 and 11.2. As a result, the mean number of entries in a Local Catalogue file was calculated at just 28% of that for a Source file due to the removal of rows and columns as discussed in Section 11.2. However, Local Catalogue files are highly variable in size, ranging from 0 to 4,541 rows, because SDSS Source files are not consistent in their size and because the proportion of “clean point sources” is not consistent from one source file to another.

One example of the way the proportion of clean point sources changes is illustrated using the `primary` flag, which *must* be set for an entry to pass the filter. This flag exists because SDSS files overlap, sometimes to a considerable degree [5] as shown, for example, in Figure 5-4. As a result, some objects will be listed twice or more. One entry for an object is designated the “primary” reference, and all others “secondary,” based on SDSS’s score for that field it is detected in – a measure of the sensitivity of the imaging system when that observation was made. [175]

For some fields in the Source Catalogue, there are no “primary” entries recorded, as all entries are included in other fields with higher SDSS scores. In these cases, no entries will be passed to the Local Catalogue. These files contain no entries, just a header which defines a “3×0” table – i.e. a table with 3 columns and 0 rows, which together amount to 5.62kB of data. In some cases, multiple fields in the same directory may produce “3×0” fields in this way, which may suggest a `run`, `rerun`, `camcol` combination which has been entirely superseded.

The mean size for a Local Catalogue file is 240 entries, and 20.2kB, however others can be as large as 4,541 rows, and 256kB. For every entry which passed this filter, the API extracted a much reduced set of only the data that would be required for the project – position (RA, Dec) and SDSS model magnitude values. (*ugriz*, designated as the better of de Vaucouleurs and exponential fits by SDSS) [111]

The average storage space used per entry was reduced by a factor of 170 due to the reduction in the number of columns. When combined with the reduction by a factor of 3.53 in the number of rows in the catalogue, this led to the Local Catalogue being 707 times smaller than the Source.

This reduction in data provides for the following advantages. Which advantage dominates depends on the particular implementation: As discussed in Section 13.2, network access dominates for the Quasar job, while for the Exoplanet job, processing and network transfer are approximately balanced

- Reduced network transfer time

By reducing file size, the time taken to transfer data from the LFC to the WN is reduced, though fixed components of this time such as authentication will remain constant.

- Reduced demands on memory

Smaller files for each field permits more of them to be accessed simultaneously, potentially allowing for larger grid jobs to be carried out on a given WN

- Reduced processing

By performing the “Extract” operation once and storing the results in the Local Catalogue, this function to separate the clean sample of point sources from the rest of SDSS does not have to be performed at every iteration of the program.

However, above all of these advantages, the most significant improvement to this project provided by the generation of the Local Catalogue was that it simplified the design of the Pipeline and provided a template for extensibility to other catalogues as described in Subsection 14.1.5.

13.1.3. Pipeline Data Metrics

The Data Analysis Pipeline (cf. Subsection 7.2.2) produced data for two catalogues: the Exoplanet Catalogue and the Quasar Catalogue. Each of these was created by means of a series of grid jobs. In both cases, the pipeline was designed to create a single output FITS file for each grid job, as described in Subsection 8.3.1.3. As a result, the data stored in these files is structured differently to that stored in the Local or Source Catalogues, and lacks the hierarchical structure of SDSS-model data. A project is on-

going to transfer this data to an SQL database to make access more convenient. Instead, output files are stored in directories based on the date upon which the grid job is executed, as shown in Subsection 8.2.5. The contents, of each of these catalogues are different, based on the different requirements of the two catalogues. Each was calibrated to optimise grid job runtimes as discussed in Subsection 9.2.1.1.

The Quasar Catalogue was produced based on a target list (i.e. the SDSS Quasar List [137].) For each entry in the target list, an individual mosaic is created from all fields in the vicinity of that target. The meant that many entries in the Local catalogue were required for each target, and File I/O and transfer time massively dominated over job processing time as discussed in Subsection 13.2.2.

In Target List mode, the output file size is directly governed by the number of entries in the target list for a given grid job. In this regard, given that each grid job listed exactly 1,000 targets, for the quasar catalogue, each of the 40 output files contains 1,000 entries. Each Quasar catalogue file is therefore the same size, occupying 87.8 kB of disc space.

The Exoplanet Catalogue, on the other hand, used the Local Catalogue in its entirety as the target list. A mosaic around each field in turn was generated by combining the neighbouring fields. Those stars in neighbouring fields were all used as potential reference stars for each Local Catalogue entry in the original field.

As discussed in Subsection 9.2.1.1, grid job runtime was limited to three days. In order to allow for a large margin of variance in job runtime, jobs length was calibrated against processing metrics developed during unit testing as discussed in Subsection 13.3.3 such that the expected mean runtime was 36 hours. In order to meet this requirement, 200 Local Catalogue files were combined to produce each grid job in this way.

The output files that comprise the Exoplanet Catalogue are highly variable in size. This is because, as discussed in Subsection 13.1.2, Local Catalogue fits files used to generate the target lists are variable in size (from 0 to 4,541 rows) because of variances in the SDSS Source catalogue. In addition, fields in a job for the exoplanet catalogue are compiled sequentially. Because of this, and since there are a number of `run`, `rerun`, `camcol` combinations which have been entirely superseded, it is possible for an entire Exoplanet Grid Job to be composed of empty Local Catalogue fields. Such a grid job

will produce an empty output table. As before, these files are completely empty except for a header describing a “6×0” data table, but still occupy 5.62kB.

The average Exoplanet Catalogue file occupies 3.22 MB and has 41,955 entries relating to stars and their pointings, if any. At the top end of the scale, the largest output file occupies 13.4 MB and consists of 175,688 entries. There are a total of 1,598 output files in the Exoplanet Catalogue.

The small number of files in the Quasar Catalogue, and their relative similarity make extraction of data from this catalogue as shown in Section 11.3 relatively simple: all 40 files can be concatenated, and various analyses performed on the whole dataset using MS-Excel.

The larger size of, and variability of the contents of Exoplanet Catalogue files makes analysis more complex. Accessing all 67,043,579 entries in 1,598 files is impractical, and concatenating these files into a single file would generate an unwieldy 5.02 GB file. Instead, for the meta-analysis carried out in Chapter 12, 17 files with 10^6 entries were selected arbitrarily from the catalogue, and used to provide a sample for analysis in MS-Excel. A future development of this project is to load the full data for the Exoplanet Catalogue into a relational database to provide for SQL access.

13.2. Processing Metrics

The production of the data at each phase of the project requires the use of a series of programs connected by a number of scripts as discussed in Section 7.2. The results of unit testing, in addition to demonstrating the functionality of the systems, as discussed in Section 9.2, were also used to identify the performance characteristics of the software.

By design, each phase of the project included a minimum work unit as shown in Table 13-2, which could not be further reduced, based on the design of the software. These work units can be aggregated to provide larger grid jobs to maximise the efficiency of the system, they cannot be subdivided between multiple worker nodes in multiple grid jobs. Estimates relating to these work units were evaluated by running the relevant software on test data on the GridUI. GridUI provided a suitable analogue for a worker node – a computer of with access to the LFC and other grid resources.

Table 13-2 lists the average time to process a work unit based on the time taken to process a small batch of work units, with locally stored data. Data Access time is influenced by both the size of and number of input files per work unit, and was overwhelmingly dominated by LFC access time. LFC Access was slowed considerably by the security & authentication requirements as discussed in Subsection 9.2.2.1.

	API (Local Catalogue)	Pipeline (Quasar Catalogue)	Pipeline (Exoplanet Catalogue)
Unit of Work	Source Catalogue file	Quasar	Field
No. of Units	421,388	77,429	358,076
Processing Time per unit	0.02887 s	0.15 s	36.0 s
Average Input per unit	1 Source Catalogue file	11.73 Local Catalogue files	20.32 Local Catalogue files
Data in per unit	11.8 MB	237.1 kB	410.5 kB
Access time per unit	4.5 s	23.5 s	40.6 s
Mean output per unit	1 Local Catalogue file	1 output file entry	240 output entries
Data out per unit	20.2 kB	90 B	18.75 kB
Ratio of Access to Processing time	156:1	157:1	1.12:1

Table 13-2: Assessment of processing elements of three phases of the project

The results of these unit tests are discussed in detail in the following Subsections. In these discussions, the dominant contribution to work unit run time is identified, and any reductions to unit run time which are possible are described. Additionally, any “economies of scale” which may be available during the grid process are identified: for example, in the case where a file which is accessed by more than one work unit, options are considered to have those work units bundled together, such that the file need only be retrieved from the LFC once.

The results of unit testing influenced the designed grid job sizes. In the case of a task for which the run time for a single unit was long, the grid job was designed to be composed of fewer work units, and vice versa as discussed in Section 13.3.

13.2.1. API Metrics

The API consists of the programs Diagnose (see Subsection 7.2.1.1), which identifies the key data columns in the source files, and Extract (Subsection 7.2.1.2), which selects

the “clean sample of point sources” entries, and copies the key data to the Local Catalogue files. These require a series of Grid Management Scripts (see Subsection 7.2.4) to download data and program files to the Worker Node, call the programs, and upload the new files to the LFC. The work unit for the API is a single file from the source catalogue, from which the output is a single file in Local Catalogue format.

Unlike with the other tasks discussed below each file to be processed in source catalogue is only accessed once in the API step. This means that there are no economies of scale which can reduce the time to access data from the LFC.

As shown in Table 13-2, this process is dominated hugely (156:1) by Data Access time. Optimisation of the processing step of this element of the project, therefore, would not significantly contribute to the overall job time.

Any improvements to unit run time must therefore come from improvements to data access time. Much of this access time, however, is dominated by LFC security and authentication, referred to as the “LFC bottleneck” in this project. Therefore, further improvements to Data Access time are outside the remit of *users* of *gLite*-pattern grid systems. [7] *Administrators* of such systems, however, may make other facilities, such as the NFS discussed in 9.2.2.1, available to users. In cloud computing, according to Fox & Gannon, the solution to this bottleneck is to move the computation to the data, where possible. [176]

In conclusion, any improvements to the unit run time for the API must focus on removing or reducing the LFC bottleneck.

13.2.2. Pipeline Metrics

The Data Analysis Pipeline, as defined in Subsection 7.2.2, consists of the Locus Algorithm Program (See Subsection 7.2.2.1) and a series of grid wrappers (Subsection 7.2.4) which manage the transfer of data from the LFC to the WN for processing.

This software operates in two modes, which have significant differences in performance and operational characteristics. These two modes are called the “Target List” mode and the “Catalogue Traversal” mode. The choice of which mode to use is governed by the

Parameterisation system used to control the project as described in Subsections 7.2.3 and 8.3.3.

13.2.2.1 Quasar Pipeline Metrics

The Quasar Catalogue was generated by the Pipeline operating in Target List mode. The work unit for this mode is the target. This meant that for each target (i.e. a quasar), a list of fields is provided by SQL query to the CAS as shown in Subsection 7.2.5. These queries list the fields that are needed to create a mosaic of files around that target. With the mosaic in place, an optimised pointing is identified for each target before moving on to the next.

As Table 13-2 shows, this process is also dominated by Data Access, with a ratio similar to that for the API of 157:1, though this similarity is coincidental. Optimisation of processing, therefore, would still provide limited benefit at this step in the process. As with the API phase, optimisation of Data Access time was not possible during the generation of the Quasar Catalogue.

However, with the pipeline operating in target list mode, there is not a direct correspondence between the two elements of input data: the target list and the Local Catalogue. In the case of the Quasar catalogue, each target required an average of 11.73 Local Catalogue files to be accessed to create the mosaic for that target. Any overlap between the fields required would create a potential for an economy of scale, as the files can be accessed once and used with multiple targets.

Multiplying the average files per target by the number of targets gives a figure of 908,242 for the number of times a field was accessed. However, as there are only 358,076 fields in the local catalogue, it follows that the mean number of times each field was accessed was 2.54, assuming all fields were accessed at least once. In principle, then, it is possible to reduce the data access overhead by approximately this factor of 2.54, by grouping together targets whose mosaics overlap.

However, developing a grouping strategy would require an additional development cycle and potentially require further data processing. A pragmatic solution to this was developed based on the fact that quasar target lists from SDSS are semi-sorted at source. At every data access phase, a checking procedure compares the file that was to be

downloaded with existing files to check if the download is necessary, which avoids the slow calls to the `gLite` system on occasions where targets in the same job have serendipitously overlapping mosaics of fields around their position.

In conclusion, as Data Transfer dominates over processing time, improvement to Data Transfer time must be emphasised. Improvements to the quasar catalogue should therefore focus on the following factors:

- As with the API, the LFC bottleneck should be eliminated if possible
- Mitigation to the LFC bottleneck may achieve a speed-up factor of up to 2.54 if all targets which share fields are grouped together
- A partial implementation of this solution was achieved by checking if a file had already been copied to the WN before accessing the LFC

13.2.2.2 Exoplanet Pipeline Metrics

The Exoplanet Catalogue was created using the pipeline in Catalogue Traversal mode. In Catalogue Traversal mode, the work unit is the field. A long target list is generated by using the Local Catalogue as a whole. Instead of a single target with a mosaic made up of multiple fields that could be included in a FoV with that target, all targets in a field are listed together, and a slightly larger mosaic is created out of the fields that are within a FoV of *any* point in the original field. Each of the targets in this field are then compared against the mosaic to create that number of optimised pointings.

In the case of the Exoplanet Catalogue, Table 13-2 shows that while Data Access remains the larger contribution to job runtime in unit testing, it no longer completely dominates, with a ratio of 1.12:1. As a result, it would be beneficial to seek optimisations and improvements to the both the processing and data access components of this phase of the project.

Improvements to the data processing time would permit the project to be iterated faster and allow more thorough exploration of the project parameter space as discussed in Section 14.1. Three main areas of optimisation are suggested and discussed in more detail in Subsection 14.3.3.2: (1) Search optimisations, which focus on improving the process by which a score is assigned to a field, which is currently an $O(n^3)$ process over

the number of candidate reference stars; (2) Grouping improvements, whereby the size of the groups of targets for which a single mosaic is generated are varied to determine the optimum group size; and (3) Balancing calculation complexity, whereby the calculations which are repeated most frequently are kept as computationally simple as possible, to minimise their impact on overall run time.

Improvements to data access time must again focus on removing or mitigating the LFC bottleneck. For each field, Table 13-2 indicates that the mean number of neighbouring fields that were required to create an appropriate mosaic was 20.32. As a corollary the mean number of times each field was accessed was also about 20.32 times. As with the Quasar catalogue, it follows that a reduction in data access time of about this factor of 20.32 compared with unit testing could be achieved by grouping neighbouring fields together, such that the fields needed to generate the mosaic around the neighbouring fields need only be accessed once.

In the Exoplanet job as implemented, grid jobs in the Exoplanet phase consist of fields listed sequentially. The Local Catalogue follows the SDSS Data structure shown in Subsection 8.2.5.1. In this structure, sequential fields form a part of a run, which SDSS defines as “a continuous scan of the imaging camera.” [158] As a result, sequential files in a run, and thus in an exoplanet grid job, neighbour one another. This means that the degree of overlap is already quite high.

As described for the Quasar job above, a “check” step was included in the Grid Management Script to prevent files being accessed through the LFC if they have already been transferred to the WN. This means that that files are accessed from the LFC less frequently than is predicted by unit testing alone.

In conclusion, as Data Transfer and Processing time are approximately balanced, improvement to either contribution to unit run time should show some improvement in the overall unit run time. Improvements to the Exoplanet catalogue should focus on the following factors:

- Improvements to Data
 - As with the API and the quasar job, the LFC bottleneck should be eliminated if possible

- Mitigation to the LFC bottleneck may achieve a speed-up factor of up to 20.32 if all targets which share fields are grouped together
- A partial implementation of this solution was achieved by checking if a file had already been copied to the WN before accessing the LFC
- Improvements to processing
 - Search optimisations, whereby the identification of the optimum pointing is accelerated
 - Grouping optimisation, whereby an optimum solution is determined for the size of the grouping of targets with an associated mosaic
 - Calculation optimisation, whereby the calculations involved in determining factors such as rating, score, or the position of a cornerpoint are simplified

13.3. Grid Metrics

Grid computing, like any distributed computing solution, cannot provide 100% scale-up efficiency: e.g. where doubling the number of computers doubles the speed. [177] Distributed computing efficiency is impaired by the fact that some central system must apportion work, distribute it to the WNs, monitor the progress of the grid job, and retrieve the results. In this case, that role is taken on by the GMS supported by the Grid wrapper software (See Subsection 7.2.4) [7] [93]

Ideally, the overhead from this management level can be minimised by creating larger grid jobs: for example, the job submission process was observed to take approximately 5 minutes, regardless of the size of the grid job. For a job that takes 20 minutes, this contribution would be 25% of the overall time, while for one that takes 1 day, the contribution would be 0.37%. However, grid jobs cannot be increased indefinitely.

In Grid Ireland, for example, the maximum permitted run time for a Grid Job was 3 days. Any job running longer than this was automatically cancelled. Large grid jobs may also require large amounts of data, and if the storage space on the Worker node is not sufficient, there may be loss of data or the job may fail.

Finally, running many jobs simultaneously may cause contention for data access resources. If there is a central data repository, either for input or output data, a large

number of jobs may cause a queue which leads to a slow down or even a stoppage in data access. This may not be apparent until the job is running.

	Local Catalogue Job	Quasar Job	Exoplanet Job
Work Units per job	1,000	1,000	200
Input data per job	11.5 GB	231 MB	80.2 MB
No. of jobs	422	78	1791
No. successful jobs	359	40	1,598
Percentage successful	85.1%	51.3%	89.2%
Output data per job	19.7 MB	85 kB	3.22 MB
Nominal time per job	1.25 hours	6.57 hours	4.25 hours
Nominal overall time	21.9 days	21.2 days	317 days
Nodes used	40	40	400
Total time taken	2.09 days	n/a	4.31 days
Observed time per job	5.6 hours	n/a	1.07 days
Effective time per job	8.40 minutes	n/a	3.85 minutes
Grid Efficiency	22.3%	n/a	16.5%
Effective scale up factor	8.91	n/a	65.6

Table 13-3: Grid metrics for the three primary grid jobs. Timing data for the Quasar job is unavailable at this time.

Table 13-3 therefore shows the number of work units per job that was chosen to provide a balance between minimising the contribution of grid management overhead to the job run time and preventing oversized jobs from causing failure.

- The nominal job lengths, as well as input and output data per job, are calculated from unit testing. These figures are based on the average work units as shown in Table 13-2.
- The effective time per job is the time taken overall divided by the number of jobs completed.
- The observed time per job is the mean processing time taken per grid job, including time spent waiting on submission or retrieval procedures, and allowing for “dead time” on nodes that were assigned but not in use.
- The Grid Efficiency is the ratio of nominal time to observed time per job, which is indicative of the proportion time dedicated to operations focussed on the actual job rather than on grid operations and wait time.
- Scale up ratio is the ratio of nominal time to effective time per job. This indicates how much faster it was possible to complete the job using the grid compared with a single standard system of the same type as the WNs.

- The number of successful jobs is the number of jobs for which output data is available. The loss of data is discussed in Section 13.4

The impact of each of these factors on the particular grid jobs are discussed in the Subsections for those jobs below.

13.3.1. Local Catalogue Job

The Local Catalogue required large-scale file I/O as it involved the manipulation of SDSS Source Data entries. As a result, the job size was calibrated for shorter jobs, such that each individual job did not place too much strain on Data Access resources such as the GMS, as fewer files were requested at the same time. The grid performance metrics for this job are shown in Table 13-3.

By choosing to group 1,000 work units (source catalogue files) together per grid job, the nominal time per job was calibrated at just 1.25 hours. Operating on a single WN, this job would therefore be expected to take 21.9 days. Instead, in the grid system operating with 40 WNs, this task took 2.09 days, a scale up factor of 8.91.

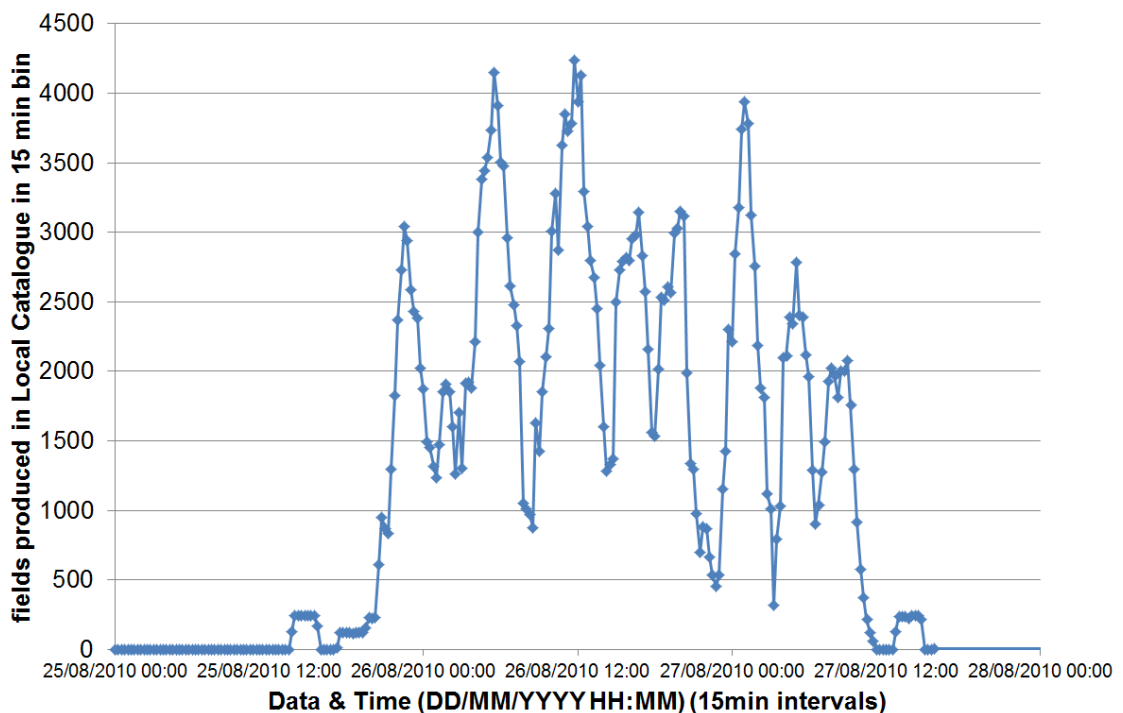


Figure 13-1: Distribution of Local Catalogue File creation times over 15 minute intervals from 25th-27th August 2010

The Local Catalogue was generated by means of a grid job that ran from 13:30 on 25th August 2010 until 15:45 on the 27th of August 2010. 359 grid jobs produced a total of 358,076 Local Catalogue files at a mean rate of 7140 files per hour, however this rate fluctuated considerably as shown in Figure 13-1, up to 4,250 in one 15 minute interval. Access to the LFC at the peak rate requests of 4.72 per second did not cause disruption to grid operations, unlike the peak access rate of 19.5 per second during the generation of the Exoplanet Catalogue as shown in Subsection 13.3.3. [178]

An irregular, cyclical pattern appears in the data. The troughs in this distribution may occur at points where jobs are either downloading files or processing files, while the peaks may be attributed to periods during which many jobs were finished processing and were uploading output. However, since the only metrics available are the start and finish times for each job, and from these the overall start, stop and count time for the overall task, it is not possible to separate processing, network download and network upload times for the grid job.

In order to further optimise the grid performance of software derived from this project, more fine-grained metrics should be incorporated into future refinements of this project as discussed in Subsection 14.3.3. These include the following

- Start and finish times for the data download phase of the grid job. Included at this step should be a track of the number of file downloads which were completed successfully
- Start and finish times for the processing phase of the grid job. Any errors or exceptions which occur during the grid job should be captured here.
- Start and finish times for the data upload phase of the grid job. A count of the number of files produced for upload should be made, and compared against the files downloaded to identify any errors in processing.

13.3.2. Quasar Catalogue Job

The production of the quasar catalogue had two primary purposes. Firstly, the analysis of the 77,429 quasars provided a list of 40,000 output results including 23,697 valid pointings as discussed in Section 11.3.

In addition, this job formed a test bed and proof of concept platform for the pipeline. In this sense, the purpose of this catalogue was to develop expertise within the SCG: for example, most Grid scripts written for the Quasar job were directly reused for the Exoplanet pipeline as shown in Subsection 7.2.4.

The quasar catalogue was generated over the course of the 5th of January, 2011. All forty files were created over a 5 minute period between 15:35 and 15:40 on that day. This narrow window indicates that all 40 jobs were launched and processed together.

Unlike with the Local Catalogue as indicated in Figure 13-1, the time taken to upload the single output file was not be a significant contribution to grid run time. Since the processing load is likely to be similar for each job, as they have the same number of targets, all jobs concluded within a short period of one another and were uploaded together.

13.3.3. Exoplanet Catalogue Job

The Exoplanet Catalogue is the primary result of this project. It consists of 67,043,579 entries and includes 61,662,376 optimised pointings for differential photometry observations on stars in the Local Catalogue, as discussed in Subsection 11.4.

Production of this catalogue demonstrates the pipeline is capable of mining large astronomical catalogues and producing large result sets. The job run time of 4.31 days as shown in Table 13-3 indicates the viability of repeatedly running the software, as required to iteratively explore the parameter space of this project as discussed in Section 14.1.

During the first attempt to run the Exoplanet Catalogue job, an issue arose whereby grid jobs overwhelmed the LFC/GMS with repeated requests. Requests to the LFC were estimated by Grid Ireland OpCentre personnel at approximately 19.5 requests per second. [178] LFC requests typically take ~2s to process and authenticate, although a number of simultaneous requests can be processed as shown in 13.3.1. As a result, LFC access was not suitable for this grid job.

Instead, this bottleneck was removed by copying the Local Catalogue to a Network File Server (NFS) as part of a LAN in TCD to which the majority of worker nodes in Grid

Ireland, but not gridUI, were attached. [93] [178] This restricted the Exoplanet Grid Job to using TCD-based WNs, and prevented detailed unit testing of access time, as the test environment, gridUI, was not connected to this LAN. However, primitive unit tests on WNs demonstrated that NFS access was over an order of magnitude faster than LFC access, thus reducing the Data Access contribution to overall job time. This made data processing the dominant factor in the Exoplanet job run time.

Additionally, to prevent overloading of the GMS, a system of job submission scripts was developed, as described in Subsection 7.2.4, to submit jobs in batches of 10 every 10 minutes. Jobs were only be submitted if there were fewer than 400 jobs running, and if fewer than 10 jobs were at a status of “Scheduled” or lower. (These statuses define grid jobs which are waiting for grid resources to become available before they can run.)

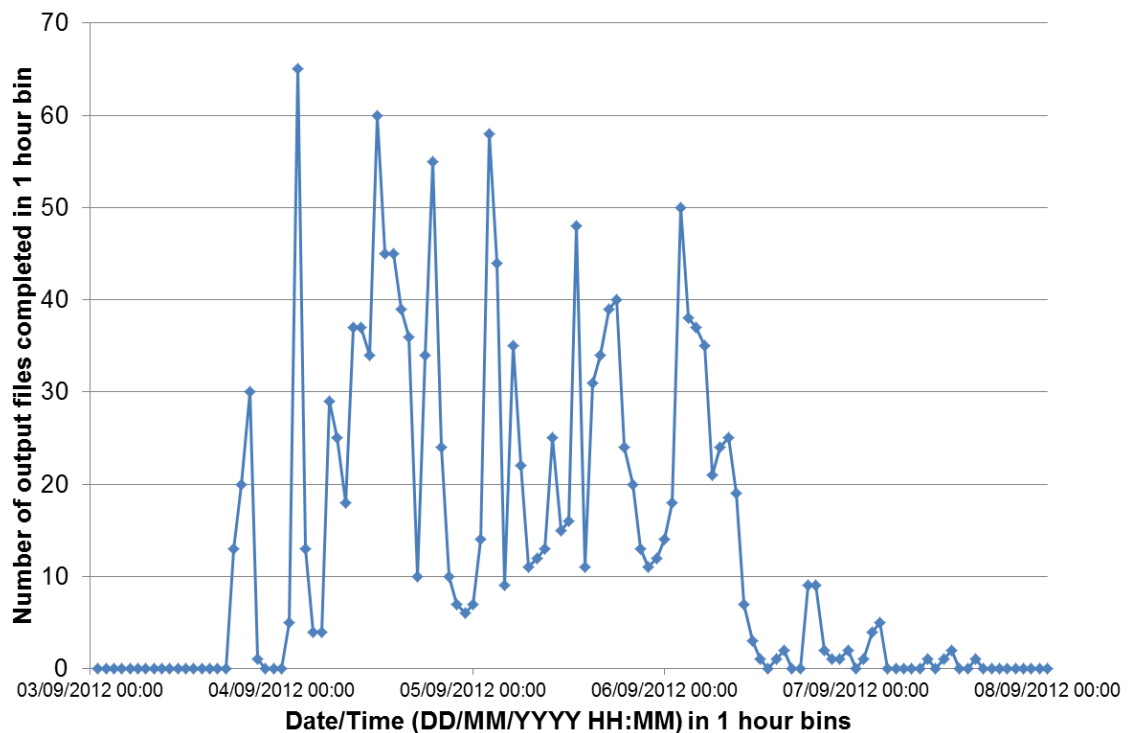


Figure 13-2: Exoplanet Catalogue output files generated over time in bins of 1 hour

The metrics shown in Table 13-3 are reflective of the Exoplanet job run under these parameters. The job was launched at 17:23 on the 3rd of September 2012 and completed at 00:57 on the 8th of September 2012 as illustrated in Figure 13-2, giving an overall runtime of 4.31 days. When compared with the nominal run time on a single WN of 317 days, this gives a scale-up factor of 65.6.

As with Figure 13-1, the peaks and troughs in the distribution of output file creation times are believed to indicate an irregular cycle between processing of jobs and writing to the grid. Far fewer output files are generated in the Exoplanet Catalogue than in the Local Catalogue, and as a result, write-based access to the LFC does not provide a significant contribution to the runtime, nor is there any likelihood of exceeding limits on simultaneous requests to the LFC due to output data.

The following can be concluded about the generation of the Exoplanet Catalogue, which may prove significant to future reuse of the project as proposed in Section 14.1

- Operations of this scale using LFC authentications are not suitable in the setup used at Grid Ireland. Some alternative *must* be available to avoid the LFC bottleneck.
- With 400 WNs, a scale factor up of 65.5 is possible, allowing for the processing of an entire catalogue traversal of the Local Catalogue in 4.31 days. This process can therefore be repeated to allow for changes to input parameters as suggested in Subsection 14.1.3

13.4. Issues Arising from Project

Prior experience at Grid Ireland emphasised complex calculations on data sets in few files, whereas this project accessed many small files and performed relatively simple calculations on each. This led to the dominance of data access time over processing as indicated in Table 13-2. As a result, the nature of this project has been described as “Data-rich, Processing-poor” [134] by analogy to the phrase “data-rich, information-poor” common to data mining literature. A number of issues arose from this nature.

A major source of the dominance of data access was the “LFC bottleneck.” The LFC is designed to permit secure access to files, but the authentication process required to provide that security takes time and occurs on a per-file basis. This became a bottleneck at all stages of the project. Data access dominated by a factor of 156:1 in Local Catalogue and Quasar Catalogue generation jobs as shown in Table 13-2, and the number of requests overwhelmed the GMS during the Exoplanet Catalogue job, mandating its replacement with a NFS-based system.

Given the dominance of data access time in the Local and Quasar catalogue jobs, a system similar to the NFS was seen to have improved the data access performance of the Exoplanet Catalogue jobs by at least an order of magnitude, and would therefore be expected to similarly improve the performance of the Local and Quasar catalogue jobs. Future projects of a similar nature to this one must use a solution to the data access problem which accommodates the data access demands as discussed in Subsection 14.3.3.1.

Data was lost in the course of this project at several points, as can be seen from Table 13-1 – not every file in the source catalogue has a corresponding Local Catalogue file, and not every entry in the local catalogue has a pointing calculated for it. However, the nature of this project means that the results are independent of one another: for example, if a particular quasar does not have an entry in the Quasar catalogue, this does not invalidate the results for the 40,000 that do.

Loss of data from the Local Catalogue does have an impact on output from the pipeline, however, this impact is similar to that which would be expected from the fact that SDSS is not an all-sky survey. For targets close to the edge of the SDSS footprint, or targets for which the Local Catalogue is missing one or more fields that should be used to create the mosaic around them; the pointing is expected to show a bias away from the absent fields, and the pointing were assigned a score worse than or equal to the score that could have been achieved with the missing part of the sky included. However, these pointings are still valid – they are the best pointing for that target that can be achieved with the data in the Local Catalogue, and their score is reflective of the reference stars that are available.

This data loss occurred primarily through one of two mechanisms: Failed Grid Jobs and Data Corruption.

Grid Jobs can fail for a number of reasons: the average success rate of user grid jobs is generally considered to be about 80%. [179] Job runtime limits mean that any job that exceeds the time limit (in this case 3 days) are automatically cancelled. This occurs when job size has been incorrectly calibrated for the job queue to which the job is submitted, or when interactive behaviour occurs during the job. Since grid, by its very nature, is non-interactive, interactive jobs will wait until they timeout. Grid jobs may

also fail because of proxy certificate expiration, due to crashes caused by coding bugs or other intermittent faults.

Resubmission of grid jobs may allow a failed job to run to completion if the failure was caused by an intermittent fault. In the case where a job fails due to exceeding runtime limits, because the job was too large for the WN to complete in the allotted time, it is unlikely that resubmission will permit the job to succeed. In this case, the job must be subdivided by the user and the sub-unit jobs resubmitted. In the current design, the subdivision of a grid job is a complex process requiring the recreation of multiple parameter files. It is recommended that subsequent projects developed from this one incorporate procedures for subdivision and resubmission of failed grid jobs.

Data corruption occurred in the course of this project when a hard disc on which part of the Local Catalogue was stored failed. This caused the loss of some files, as backup was not available for the scale of data used in this project. These files were then unavailable for the generation of the Quasar and Exoplanet Catalogues. Future projects developed using cloud computing as discussed in Subsection 14.2 may be able to avail of cloud storage solutions such as Amazon Web Services' S3, which claims 99.999999999% data reliability, which should prevent loss of data via this route. [180]

Data can also be lost due to user action. During routine “clean-up” operations on gridUI, the logfiles for the Quasar Catalogue generation job were deleted without backup. As a result, detailed metrics on that job are unavailable, as discussed in Subsection 13.3.2.

Finally, data may be discarded by design. The creation of the Local Catalogue greatly reduced the data size from the SDSS Source Catalogue. This permitted a much smaller data communications overhead. However, in reducing the data per entry from 14.6 kB to 86 B, some data that may have been desired was discarded. For example, ObjectID, a unique identifier for each object in SDSS, was not retained in the Local Catalogue. Future projects may consider including more data from their source catalogue as discussed in Subsection 14.3.2.2.

Unit testing is a powerful tool, but limited in its application. Unit testing does not reveal scale-up issues, such as the LFC shutdown as discussed in Subsection 13.3.3. Repeated

full-scale testing on the same data takes time, and may be expensive, but some full-scale testing is needed to fully characterise the distributed computing solution. A pragmatic solution may be that when creating additional catalogues as discussed in Section 14.1, it may be possible to use these jobs as full-scale tests of the software system

13.5. Conclusions

The Locus Algorithm software, defined in Chapter 5 and designed in Section 7.2 provided a software solution to the Astrophysical challenge, under the conditions of square fields of view limited to North-South/East-West translations. It consisted of two primary software components, the API (Subsection 7.2.1) and the Pipeline (Subsection 7.2.2.)

The API extracts data from the SDSS Source files to create the Local Catalogue. In unit testing, this element was dominated by Data Access time. A work unit for the API was identified as the source catalogue file, and was estimated from unit testing to take an average of 4.5 seconds to process.

The Pipeline operated in two modes: target list mode (used to generate the Quasar Catalogue) and batch mode (used to generate the Exoplanet Catalogue.) Target list mode was, like the API, dominated by Data Access time. The work unit was determined to be the individual target (i.e. a quasar) which was estimated to take 23.5s to complete. In testing Batch mode on work units, each consisting of a Local Catalogue field, the job time per work unit was determined to be 40.6s.

The grid implementation of both the API and the Pipeline in target list mode bundled 1000 work units into grid jobs, giving grid job run times predicted at 1.25 and 6.57 hours respectively. The API job runs took 5.6 hours on average, including time spent in the queue. The entire Local Catalogue was complete in 2.09 days, representing a grid scale up factor of 8.91. Metrics for grid performance are not available for the API in target list mode.

The initial attempt to use the Pipeline in batch mode for the creation of the Exoplanet Catalogue failed, but in doing so revealed a major bottleneck in data access times: i.e. the LFC. An alternative was implemented whereby data was stored on an NFS attached to the worker nodes via LAN. With this alternative in place, 1,791 grid jobs consisting

of 200 work units, predicted to take an average of 4.25 hours were submitted. 1,598 of these jobs were completed over 4.31 days on 400 WNs. This indicates an average job time, including queuing, of 1.07 days. Taken overall, a speed up factor of 65.6 was achieved with 400 computers.

Three catalogues were generated over the course of this project, the Local Catalogue, the Quasar Catalogue and the Exoplanet Catalogue. The Local Catalogue, as designed, reduced the amount of data that must be transferred between nodes and LFC by a factor of 707. This permitted repeat access to the data stored within the Local Catalogue without requiring regular large-scale data transfer as would have been required, had the Source Catalogue been used for the pipeline instead. The Quasar Catalogue has 40,000 entries, including 23,697 pointings in 40 nearly-identical files of 1,000 entries each. The Exoplanet Catalogue, on the other hand, includes 67,043,579 entries and 61,662,376 pointings. Files in this catalogue ranged from 0 to 175,688 entries.

Several lessons for future projects were discovered. First, any future project must not be dependent upon the LFC or a similar slow-access system for time-sensitive data. Second, Data loss and corruption must be minimised, by choosing a high-reliability data management system. Finally, future development of the testing regime must involve the capability to test or simulate the impact of grid systems on job run time.

14. Future Use & Refinements of the Project

This Chapter considers future work for this project under two headings: Reuse, which refers to ways the software solution to this project can be reused in a largely unchanged form and Refinement, which is used to describe changes to the underlying functionality of the project. Both reuse and refinements to the project would be assisted by migration of the software system to a new, more modern HPC paradigm, for example Cloud Computing. Although these headings are discussed separately here, it is envisaged that future work based on this project would incorporate several or all of these suggested expansions.

This project has developed a system capable of analysing large astronomical catalogues to generate optimised pointings for particular sets of targets under particular sets of observational requirements. Two specific target sets have been analysed, producing two output catalogues as mandated in Subsection 4.1.2 – a Quasar Catalogue and an Exoplanet Host Candidate Catalogue. It is possible to reuse this system to create new catalogues for other target sets, or under other observational criteria. Section 14.1 therefore discusses options for the exploration of the full range possible for input data, and provides suggestions for the relative utility of these options.

Section 14.2 discusses the suitability of the project for migration to a Cloud computing paradigm. A move toward a cloud solution is made all the more relevant by the closure of Grid Ireland, the High Performance Computing Centre upon which this project, and software solution, was dependent. An assessment of the conceptual similarities the Amazon Web Services and the gLite grid system provides an illustrative example of how this might be implemented.

The core algorithm of the project was required to be computationally simple. This requirement imposed boundaries on the utility of its output. The first refinement to the project which is proposed in Subsection 14.3.1 is to incorporate changes which remove or mitigate those boundaries.

The scoring system developed in Section 5.3 provides an estimate of the degree to which a set of candidate reference stars show similar colour to a given target. Subsection 14.3.2 proposes a system by which the scoring system can be calibrated, and

suggests improvements to that system by means of incorporating additional variables into the score, or applying a non-linear system for combining ratings into scores.

Chapter 13 posits a number of conclusions about the computational performance of the software system developed and the High Performance Computing solution implemented in this project. Subsection 14.3.3 builds upon these conclusions by recommending a number of ways by which Data and Processing performance might be optimised.

14.1. Reuse of the Project

Over the course of this project, the catalogues generated were focussed on particular purposes – the Quasar and Exoplanet jobs. To do this, as discussed in Chapter 11, input parameters were selected to suit those purposes. As a result, the exploration of the possible parameter space – the range of possible values of input data – with the software was relatively narrow.

By reusing the software solution developed in this project, it is possible to explore more of the range of possible input data, parameters and arguments for the Locus Algorithm, by changing some or all of the following input data

- Magnitude and Colour variation limits, and choice of SDSS magnitude band. As shown in Chapter 12, these factors contribute directly to the score for a given target. As such changes to these, as discussed in Subsection 14.1.1, are predicted to cause significant changes in output.
- Observational parameters such as FoV size and Resolution, which vary from observatory to observatory, allowing for the generation of output catalogues suitable for telescopes of various capabilities as described in Subsection 14.1.3
- Target parameters, such as using the system in target list mode for objects of a type which have not yet been tested such as variable stars as proposed in Subsection 14.1.4
- Data Source: by making modular changes to the API, as suggested in Subsection 14.1.5, it is possible to use the software system with other existing catalogues or new catalogues as they become available, which may cover magnitude, spectral or positional ranges which SDSS does not.

This exploration of the parameter space may provide additional value to the project. To this end, it is proposed to examine the impact of varying the parameters on the results of the project. This impact may be assessed in a manner similar to the meta-analysis of the catalogues as discussed in Chapter 12 – primarily by observing the change in the distribution of score as parameters vary.

Note that an increase in score due to variation in parameters does not necessarily indicate a more suitable target or reference star – the normalisation in the generation of ratings means that scores are only comparable within a given catalogue. As a result, variation in each parameter adds another dimension to the “refinement” task of characterising the scoring system discussed in Subsection 14.3.2.

14.1.1. Magnitude and Colour Arguments

There are three parameters in this project based on SDSS magnitudes, as discussed in Subsection 7.2.2.1: maximum permitted magnitude difference, maximum difference in colour index and thirdly, the SDSS filter band for which the catalogue is designed.

As designed, maximum magnitude difference is treated as a binary limit – stars are either accepted as potential reference or rejected based on whether their magnitude is within the limit of that of the target. Therefore it is expected that increasing the limit would increase the number of potential reference stars, and thus increase the score of a given target.

This increase in score is predicted to be, on average, directly proportional to the increase in the *number* of stars available as references. However as shown in Figure 12-1, the number of stars in SDSS does not bear a linear relationship with magnitude, and as a result a given change in the magnitude difference limit would not be expected to show a linear response in score.

As defined in Section 5.3, the rating for a given candidate reference star is calculated by measuring the difference between the colour indices of the reference and those of the target. Those candidates for which this difference exceeds the maximum colour difference given as an argument are excluded from consideration. Candidates that are not excluded are assigned a rating between 1 and 0, with 1 identifying candidates with

an exact match to the target in a given band, and 0 indicating targets with a colour index difference exactly equal to the maximum permitted.

Therefore, changes in maximum permitted colour index difference between target and reference stars would be expected to cause substantial changes in ratings, as illustrated in Figure 14-1. Since ratings are combined into scores and used to identify optimum pointings, any changes to the maximum permitted colour difference would be expected to have a significant impact on the score for a target.

Increasing the maximum permitted colour difference would increase the number of potential reference stars, and increase the rating for the existing reference stars. As a result, increasing this limit is predicted to lead to an increase in score. However, colour indices for stars in SDSS have been measured to be multi-modal as illustrated in Figure 12-5. This multimodal behaviour, combined with the complex relationship between rating and colour as illustrated in Figure 5-11 and described as the “Witch’s Hat Pattern” in this project prompts the requirement of further, detailed investigation to determine their impact on *Score* for given fields.

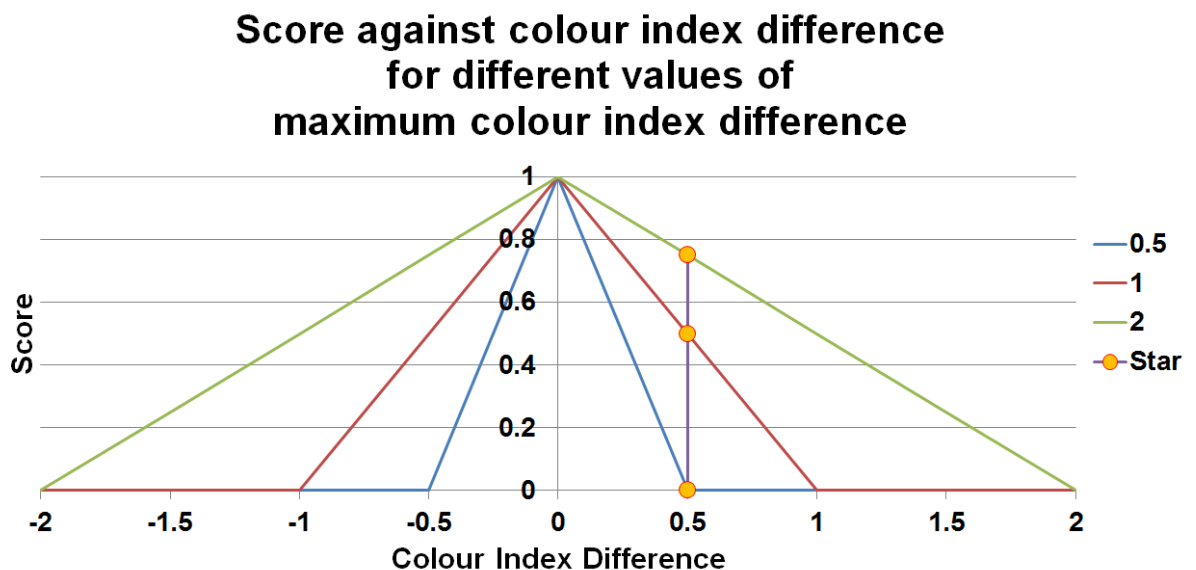


Figure 14-1: Effect of changing maximum colour index difference between target and reference star on the rating. A candidate reference star with a colour index difference of 0.5 is shown as an example. When the maximum permitted colour difference ($\Delta\text{Col}_{\text{max}}$) is 0.5, its rating is 0. When $\Delta\text{Col}_{\text{max}}$ is 1, its rating is 0.5, and when $\Delta\text{Col}_{\text{max}}$ is 2, its rating is 0.75.

14.1.2. Choice of SDSS filter

Concerning the impact of spectral filters on *Score*, it is certain that choosing a different spectral filter for a catalogue would change the scores and pointings for individual stars, and this have an impact on the overall distribution of scores.

As an example on further investigations in this regard, the filter used for both catalogues generated in this project was the SDSS *r* filter alone, suggesting that exploration of the *g* and *i* bands would readily test the system under sufficiently similar conditions with regard to how different filters affect the rating of stars and their associated scoring of fields that a comparison between *g*, *r* and *i* would be possible.

In like fashion, examination of the more extreme *u* and *z* bands of SDSS [43] (where there is only one neighbouring filter against which colour index can be calculated) could be investigated on whether, as expected, fewer stars would be rejected as being outside the limit of maximum colour difference.

The calculation of rating for a star in this band would use the linear rating system shown in Figure 5-10, which has a mean for a random distribution of 0.5 as opposed to the Witch's Hat pattern illustrated by Figure 5-11 which has a mean for a random distribution of 0.25. As a result of fewer stars being excluded, and a higher mean rating, substantially higher scores can be expected from catalogues based on analysis of the *u* and *z* bands.

14.1.3. Observational Parameters

The two observational parameters submitted as arguments to the software, as defined in Subsection 7.2.2.1, are field of view size (FoV) and resolution. These parameters are passed to the Locus Algorithm program as command line arguments. This makes them amenable to changes at a user, rather than developer level.

14.1.3.1 Field of View Size

The field of view size is the length in degrees of the side of the square that forms the usable area of the field of view. This value is calculated based on the particular combination of telescope and camera at a given observatory. This means that each catalogue is optimised for a particular observatory, and any observations made at a

different observatory are potentially problematic, unless the two observatories use similar equipment.

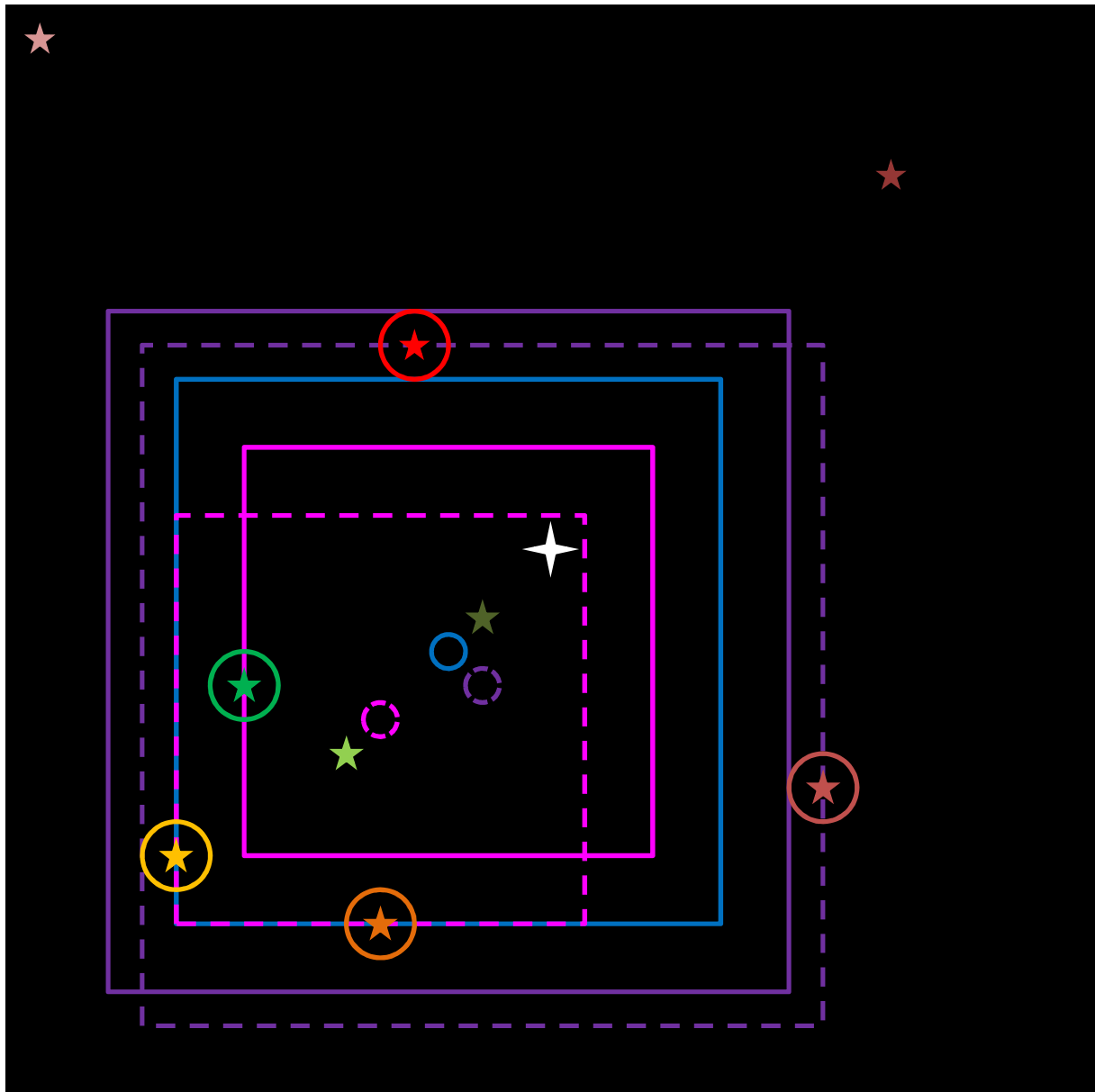


Figure 14-2: Locus Algorithm with different FoV sizes. Target, references and pointing identical to Figure 5-8. Blue outline shows the original pointing. Purple and Magenta show larger and smaller FoV sizes respectively. Solid lines show results of using the originally calculated pointing. Dashed lines show the optimal pointings for those FoV sizes. Circled stars indicate those references affected by the size of FoV

The Locus algorithm, by definition, places at least two stars at the edge of the usable field of view. This means that observations made with a narrower FoV than that for which the catalogue was optimised automatically lose two, and possibly more,

designated reference stars, as shown in Figure 14-2, with the magenta squares,. As a result, the photometry for the smaller field may be compromised, and therefore pointings optimised for a FoV larger than that of the telescope at the observatory should not be used.

On the other hand, if an observatory with a larger FoV uses the output of a catalogue optimised for a smaller FoV, the FoV will automatically include all of the reference stars included in the smaller one. However, the larger FoV may include stars which would have been viable references, but which have been excluded from consideration by the algorithm for the smaller FoV. In addition, it may have been possible to reposition the larger field to include even more references as illustrated with the purple squares in Figure 14-2.

It may therefore be concluded that in the absence of a perfectly optimised catalogue, the pointings optimised for a smaller FoV may be used, and will work at least as well on a telescope with a larger one, though they will not be optimised for that telescope.

It is therefore desirable to explore more of the FoV size parameter space to allow for observatories with many sizes of FoV to benefit from the Locus Algorithm's optimised pointings. In the context of use of this system by an individual observatory to generate optimised pointings for a particular target list, it is advised that the observers take particular care to select a FoV parameter corresponding to the usable area of their detector, which is smaller than the edge-to-edge size.

14.1.3.2 Resolution Parameter

As defined in Subsection 5.2.2, the resolution parameter defines a lower limit for separation of point sources which is again dependent on the observatory for which the catalogue is to be generated. Contributions to this limit include the Point-Spread Function (PSF) for the telescope, the pixel scale of the camera, and atmospheric effects such as seeing. Stars whose separation in RA and Dec is less than this value are excluded from consideration as reference stars.

If the resolution for the telescope for which a catalogue is to be generated is poor (i.e. the resolution parameter has a high value) then the separation at which stars are to be excluded increases, and as a result the number of stars excluded increases.

If there are many objects in a field the likelihood of there being overlap between two or more stars increases. If the mean separation between stars drops close to the resolution parameter, this is known as a “crowded field.” In a crowded field, resolution may become a dominant factor, as more and more stars are excluded from consideration as references. Therefore, a field with a high score when generated for an observatory with good resolution may be crowded when used with a telescope with poor resolution.

On the other hand, any observatory with better resolution than Apache Point Observatory (APO) – the observatory used in the generation of SDSS – will be able to resolve all resolved stars in SDSS and more besides. This means that a catalogue generated for such an observatory may miss out on candidate reference stars which APO could not resolve but which that site could. Catalogues generated for such observatories cannot, therefore, be fully optimised based on a catalogue generated from SDSS.

Further examination of this parameter would prove valuable for sites with resolution *poorer* than that at APO, but not for sites with greater resolution. As with FoV, users should take particular care to enter the resolution parameter that pertains to the observing conditions for which they intend to generate optimised pointings.

14.1.4. Target List Options

This project operates on the basis of input data in two forms – target list and the local catalogue, which contains the potential references. For each target in the target list mosaics are excerpted from the Local Catalogue files surrounding those targets. These are then used as a list of candidate reference stars.

The catalogues created for this project used two target lists. The Quasar Catalogue used the list of all known quasars in SDSS as a target list. The Exoplanet Catalogue used the Local Catalogue itself as the target list, and compared it against itself to produce optimised pointings.

It may be possible to expand this parameter space by providing another target list, for example, variable stars. This would allow the pipeline to be operated in target list mode, and generate a new catalogue similar to the Quasar Catalogue, listing optimised pointings for the new targets. For any set of variable targets in the SDSS footprint, it is

possible to generate a new catalogue of optimised pointings for differential photometry by creating a new target list and running the pipeline in target list mode. No significant changes need to be made to the system to provide for a changed target list.

14.1.5. Data Source Parameters

As discussed in Section 2.3, SDSS is the chosen catalogue for this project because at the time the project was initiated, it was the largest photometric survey in the optical range. [84] However, SDSS sky coverage is incomplete, even in the most recent data release, DR12. [175]

Other catalogues have other advantages. Some offer wider sky coverage: for example, the United States Naval Observatory (USNO) catalogues, including USNO-B: an all-sky catalogue believed to be complete down to $V=21$. [181] Others are able to observe fainter objects – the forthcoming Large Synoptic Survey Telescope (LSST), for example, is expected to be able to map objects as faint as $r \sim 27.5$. [85]

As shown in Subsection 7.2.1, the API provides a layer of data abstraction, by providing a template for generating a Local Catalogue from any existing catalogues. It is possible to develop an API suitable for importing data from other sources such as LSST into Local Catalogue format. This can be achieved by editing modular components of the Diagnose and Extract programs to suit the structure of the new source catalogue.

Because of this abstraction, the pipeline should be able to operate on this new Local Catalogue with a minimum of modification – ideally, none. Any new catalogue generated in this manner would require a new series of meta-analyses modelled after those in Chapter 12 to characterise its behaviour.

14.2. Migration to Cloud

Cloud Computing is a HPC solution which was in its infancy when this project was under development, but which has since become a mature technology. Most Cloud implementations are based on the concepts of Infrastructure- Platform- and Software as a Service (IaaS, PaaS and SaaS) whereby the user interacts with remote services via an internet connection, and pays for those services on a per-use basis. [177]

As cloud technology may be considered an evolutionary development of grid computing, it may be possible for this grid computing project to evolve into a cloud computing one. [88] In order for this project to migrate to any new platform, that platform must replicate or replace each of the following core components of the existing HPC solution, as discussed in Subsection 3.2.1

- User Interface

Any system used to replace the Grid Ireland system used in this project must allow for an interface by which the user may upload data, initiate and monitor jobs, and retrieve results. In the grid paradigm, this function is handled by gridUI

- Storage

This project relied upon access to 10 TB of storage for access to source data (4.76TB), the Local Catalogue (6.89 GB), and Output Catalogues (up to 5.02GB) as provided by the LFC. A replacement solution must provide similar capacity, and ideally would have a more robust system for data access with greater capacity for many simultaneous access requests.

- Apportionment

Much as in the current grid system, the Parameterisation program defines grid jobs, and the Job Submission System (JSS) apportions those jobs to Worker Nodes (WNs), any new solution which is provided must incorporate a system for apportioning work over the new distributed execution system

- Execution

The WNs in this project acquired the data from the LFC, but then executed the grid job by running standard programs which could be developed and tested on conventional hardware without reliance on grid architecture for unit testing, which rapidly accelerated the development cycle. Any solution to which this system can be readily migrated should include a similar system for the execution of distributed jobs which functions in a manner similar to a classic, serial computer.

Migration to a Cloud computing paradigm is made possible by exploiting the modular nature of the software solution and replacing grid-specific modules with corresponding

cloud-based systems. It is possible to demonstrate the close correspondence between the existing system and a cloud implementation by using the Amazon Web Service (AWS) as an illustrative example.

AWS is a cloud computing system developed by Amazon to allow third party users to rent the use of distributed storage and computing facilities, amongst others. [182] Figure 14-3 illustrates how AWS may be used in a similar conceptual manner to that in which the grid is used in this project as was shown in Figure 3-3. This similarity of design makes this project particularly amenable to conversion to AWS.

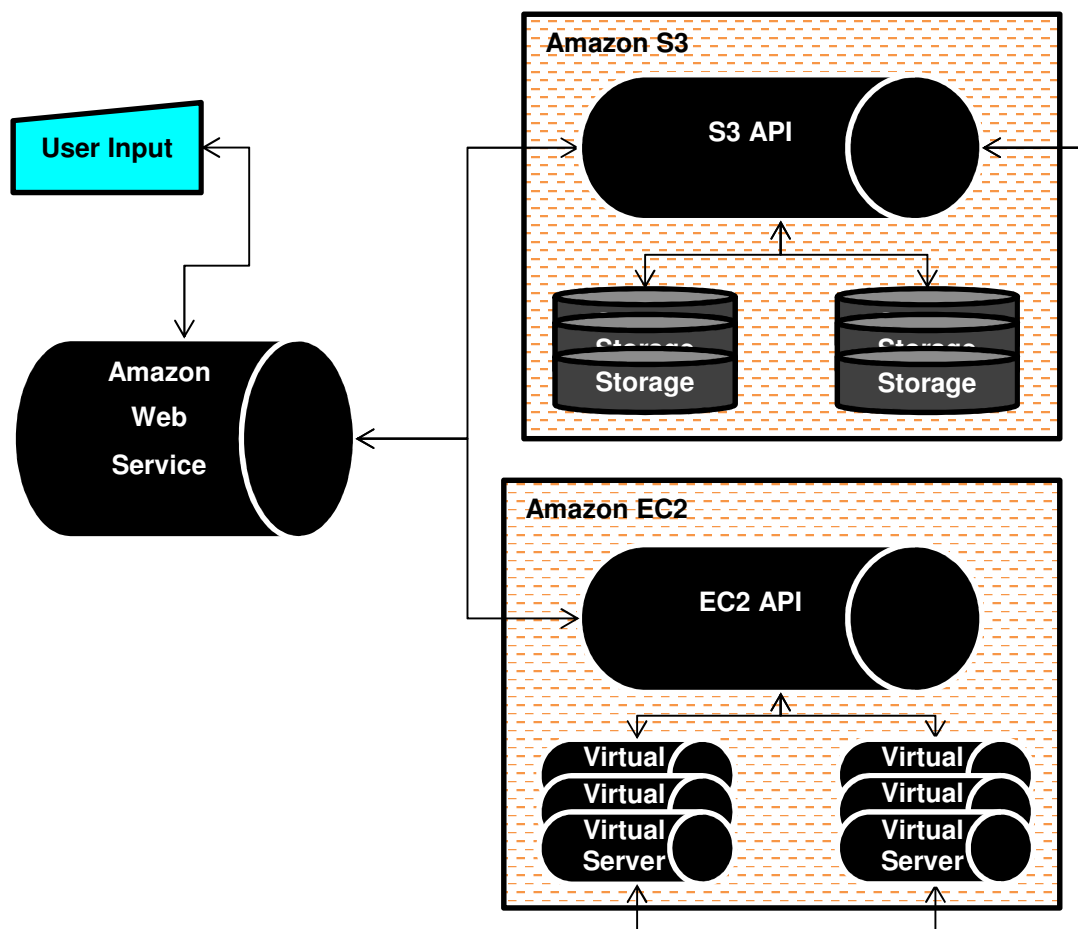


Figure 14-3: Conceptual structure of the AWS from the user perspective. The similarity of this structure to the structure of the grid as shown in Figure 3-3 suggests that many components of the software could be reused should the project be translated into a cloud computing paradigm

- User Interface

Access to the AWS is provided through a web front end, which provides a Graphical User Interface to the various AWS systems, including storage,

resource apportionment and job execution, as well as additional tools such as developer tools which allow for software development and deployment. From this website, cloud jobs can be managed just as grid jobs were managed from gridUI.

- Storage

The Amazon Simple Storage Service (S3) is a distributed storage system wherein data is stored redundantly on multiple physical servers. [183] Data may be uploaded to S3 using the web interface, which supports parallel uploads from multiple clients. [184] Data on the S3 may be accessed directly by other AWS systems. This data is stored as objects in containers called buckets, which form trees analogous to the directories in a file system. [182] Data in S3 is considered immutable, and changes in that data take time to propagate, which makes it unsuitable for interactive use. [182] For these reasons, S3 may be considered analogous to the LFC in Grid Ireland.

Coincidentally, a subset (~5% - 180GB) of SDSS DR6 is available as a sample public data set on S3, which may prove useful in testing the cloud implementation of this project. [185] In principle, optimised cloud computing design allows for the movement of programs to the storage location, rather than the other way around. [177] If implemented correctly, this would avoid the data I/O bottleneck highlighted with the use of the LFC in this project.

- Apportionment

Elastic Cloud Compute (EC2) is the computing element of the AWS. [183] It operates by creating and running virtual machines (VMs) on demand. [182] [183] These virtual machines can each be assigned particular elements of the overall computational task by EC2 in the same way that WNs on the Grid were assigned tasks by the GMS.

- Execution

The VMs generated by EC2 may be considered analogous to the physical WNs on the grid. Much as each WN had access through the `gLite` interface to the LFC, each VM has access to the data stored in the S3. When apportioning resources to the VMs, the user can select from a series of options regarding the virtual server including memory, storage, OS etc, in a manner analogous to the

way grid job requirements could be set using JDL . [183] The OS options include UNIX systems similar to those used in the grid implementation of this project, which should simplify the migration process. [184]

Although the detail of how AWS operates diverges significantly from that of Grid Ireland, notably in its use of VMs instead of physical WNs, the similarities in conceptual design are illustrated above.

This means that this project would not need a fundamental change in design in order that a migration be viable. A proposal to carry out this migration in collaboration with DIT School of Computing has recently been put forward. [186]

14.3. Refinement of the Project

While reuse of the project, for the most part, requires user-level changes in parameters and input data, more significant refinements of the system may require modifications at the program level. Some of these changes are relatively simple changes that are described in full here, while others demand a more extensive redesign and development process. In the latter case, a brief description of the design requirements is given here.

Three areas of refinement are highlighted – the first two are driven by astronomical requirements and the last by a desire to improve computational performance. These areas are broken down as follows

- Expanding the Locus Algorithm to remove certain limitations on its current functionality, such as
 - Errors which occur near to $RA \approx 0^\circ$
 - Approximations in the coordinate system which are inaccurate near the poles
 - Allowing for the rotation of the field of view of the telescope, in addition to the translations permitted in the current algorithm
- Refining the scoring system based on experimental evaluation, perhaps by incorporating additional parameters to the calculation of score, or by applying a sliding scale for the value of potential references after the first reference

- Computational Optimisation of the software solution by implementing improvements to the Data I/O and Data Processing time

14.3.1. Addition of Functions to the Algorithm

The current Locus Algorithm as discussed in Chapter 5 provides optimum fields for observation using square CCDs locked on a North-South/East-West orientation. It provides for translation of this field by adding or subtracting half the size of the field of view from the RA and Dec coordinates of each of the reference stars in turn. The RA term is corrected for spherical effects by dividing the adjustment by the absolute value of the cosine of the declination of the target.

These operations are computationally very simple, but this simplicity imposes limits on the functionality of the algorithm. In future expansion to the project, the additional functions specified in the following Subsections may be added. When implementing these functions, care must be taken to balance increased functionality with increased processing time.

14.3.1.1 RA \approx 0° Functionality

The first limitation is that the simple addition/subtraction technique ignores the fact that RA “wraps around” at RA \approx 0°. This means that when adding boundary boxes (cf. Subsection 5.2.3) or counting whether a target is within half a field of view of a pointing (cf. Subsection 5.2.5), stars at RA = 359.9° are treated as widely separated from targets at RA = 0.1°.

Note that for the small FoV used in this project (0.166° and 0.25° for the Quasar and Exoplanet catalogues respectively), this limitation only impacts a small strip of the sky around RA \approx 0°. Additionally, the SDSS Legacy survey upon which the data for this project was based was focussed on the North Galactic Cap, with a distribution centred at about RA \approx 180° [111], diametrically opposite to the region in which this issue arises. Out of 585,634,220 entries in SDSS, just 1,305,826 [6] entries are found within 0.25° of the RA=0° line. This represents just 0.223% of the overall catalogue. As a result, this limitation does not impact a large proportion of the results generated in this project.

In addition, as with the case of missing Local Catalogue files, or for stars near the edge of the SDSS footprint as discussed in Section 13.4, the pointings for stars near $RA \approx 0^\circ$ are likely to be biased away from the missing candidate reference stars, but remain viable pointings with usable references, even if those references do not include all possible references.

However, a refinement which would improve these results would be to incorporate a check to identify if a particular operation passes the first point of Aries should be implemented at these two phases, and 360° added or subtracted as appropriate to ensure comparability between neighbours to allow for general application of the software solution.

This function is computationally simple and of constant time. As a result, the impact on overall runtime for the project should be negligible.

14.3.1.2 Polar Fields & Large Fields

As discussed in Section 5.1, the use of a square detector and a catalogue defined in polar coordinates demands correction for spherical effects – an east-west line of a given angle of RA does not subtend the same true angular length at different values of Dec except at the celestial equator. The relationship is given by Equation 14-1.

$$angle\ in\ RA = \frac{true\ angle}{\cos(Dec)}$$

Equation 14-1: Relationship between true angle and angle in RA

However, trigonometric functions are computationally intensive, especially when iterated repeatedly as in this project. [187] As a result an approximation was used whereby the same correction factor was used for all stars for a given target, by dividing by the cosine of the Dec of the target instead of the individual references.

For small fields, such as those used in this project, this approximation is very close outside the polar regions. (The difference between the correction factor for the upper and lower edges of a 0.25° field such as those used in this project is just 1% at $66.5^\circ N$.)

Within the polar region, and especially at $Dec > 80^\circ$, this effect becomes much more significant. As a result fields identified by the Locus Algorithm in the polar regions are

less reliable as shown in Figure 14-4. The variance between upper and lower edges of the FoV is also more pronounced for larger FoVs.

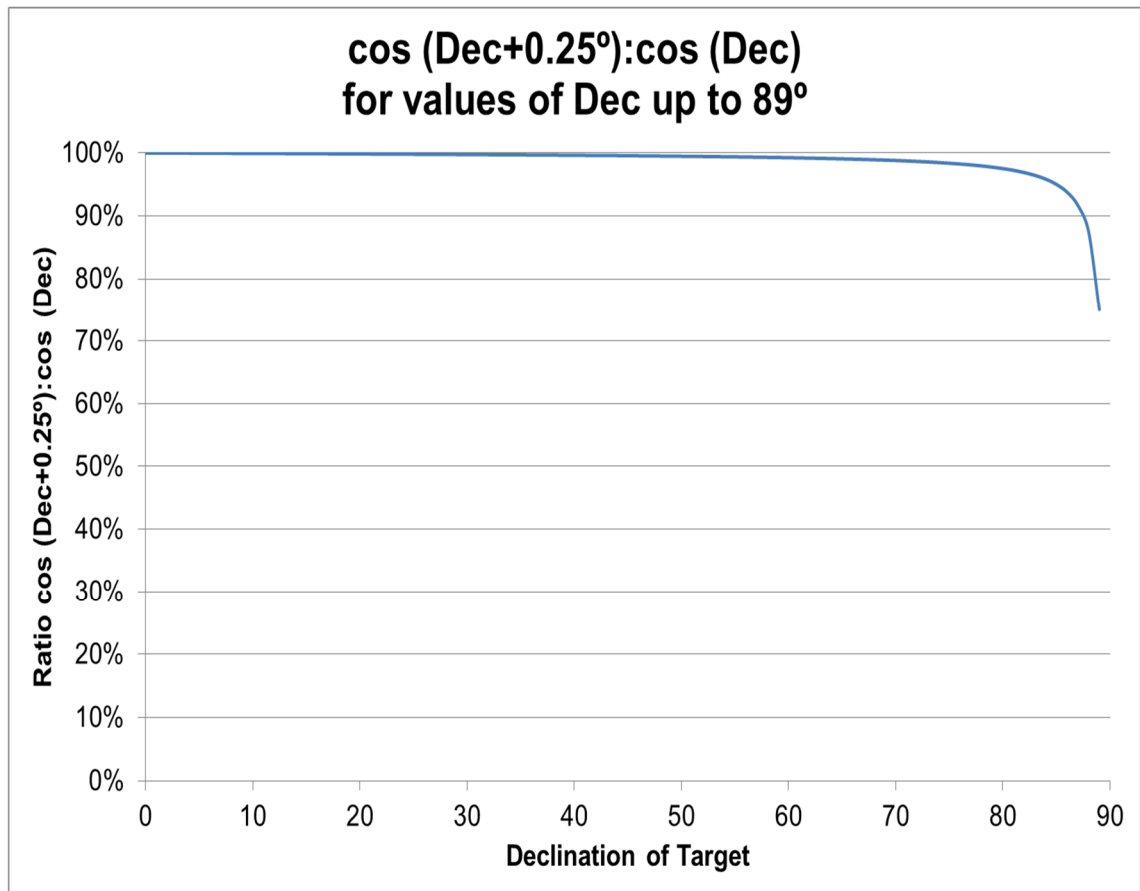


Figure 14-4: Ratio of the spherical correction factor for the northern edge of a 15arcminute FoV to that for the southern edge. As is shown, these values are very close to equal at low-medium Dec.

Given the focus of the SDSS Legacy survey used in this project on the North Galactic Cap, very few stars are observed in the source catalogue North of Dec = 66.5° (1.884%) and none are observed near the Southern Celestial Pole. [111] The impact of this limitation on the output catalogues generated in this project is therefore small.

To enable the software to accommodate these regions would require that independent spherical corrections be calculated for each star. However, as indicated by Hindriksen, 2012 [187], cosine calculations can be about 60 times as intensive as addition or subtraction. This means that, should this solution be implemented, it must be done carefully to minimise iterations of this function. See Subsection 14.3.3 for more information on suggested optimisation strategies.

14.3.1.3 Allowance for Rotations

Rotating fields of view were not included in this project: the camera is assumed to be mounted to the telescope on a locked North-South/East-West axis. The pointings and FoV generated are thus best suited to use in an automated astronomical survey, where the telescope system is mounted in such a way that telescope translations on those axes are possible, while rotations about the length of the telescope are not permitted.

As with using a catalogue for a smaller FoV with a larger one as discussed in 14.1.3, the output catalogues of this project are still useful, if not 100% optimised, for manual systems, and automated systems in which rotating FoV are permitted. Firstly, the pointings generated provide an improved set of reference stars to a direct pointing as shown in Figure 14-5, and may be used as a base from which manual adjustments may make improvements. Secondly, by providing a catalogue of pointings and scores, stars which serendipitously have a large number of very similar neighbours are highlighted. These stars will still have at least as many candidate reference stars in a rotated FoV as in a locked one.

The reason that rotations are not accounted for in this implementation is the computational complexity of rotational calculations, especially given the need to correct for RA as Dec increases as discussed in Subsection 14.3.1.2. However, as shown in Figure 14-5, rotation may be useful to further optimise the FoV for a given target.

A preliminary proposal is to expand upon the existing system based upon the guiding principle of the Locus Algorithm. The Locus Algorithm is based on that premise that the only points in the sky that need be considered as possible pointings for the telescope are the points of intersection between the loci traced by the centre of a field of view that will just include a given target. It is at these points that the number of stars that can be included in the field changes.

The Locus Algorithm works by defining North-South/East-West boxes around stars, then identifying and considering the points of intersection between those boxes. In the Locus Algorithm, a series of pairwise comparisons with other candidate reference stars are made to identify the points of intersection between those boxes.

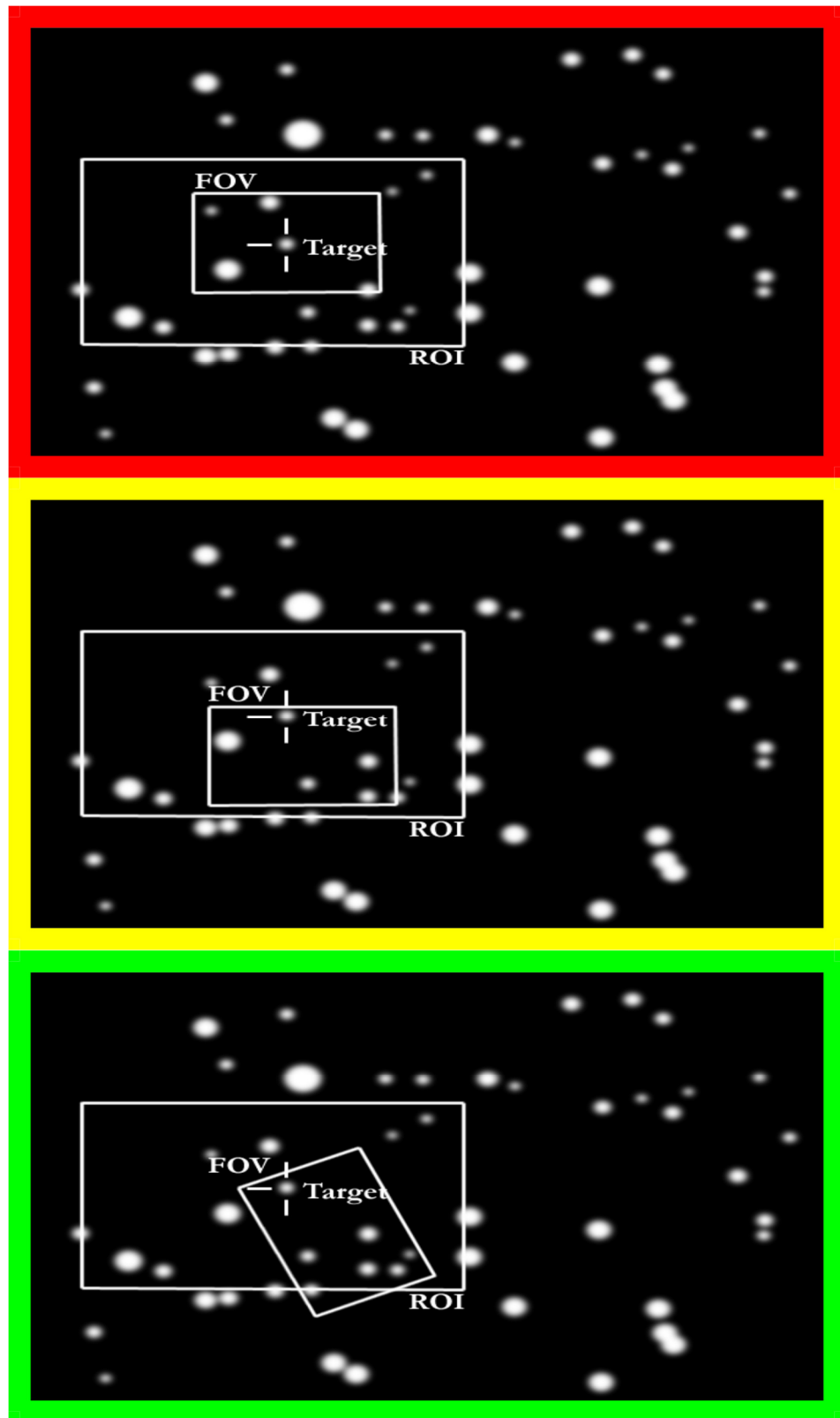


Figure 14-5: Position and orientation of the Field of View (FoV) can maximise the number of reference stars. Images: Stephen O'Driscoll, Dept. of Applied Physics & Instrumentation, CIT. [2] (Duplicate of Figure 2-9)

The expansion (referred to as the “Spiderweb Algorithm” after the weblike pattern of lines generated) applies this same principle to the orientation of the FoV – the only orientations that matter are those where the number of available reference stars changes.

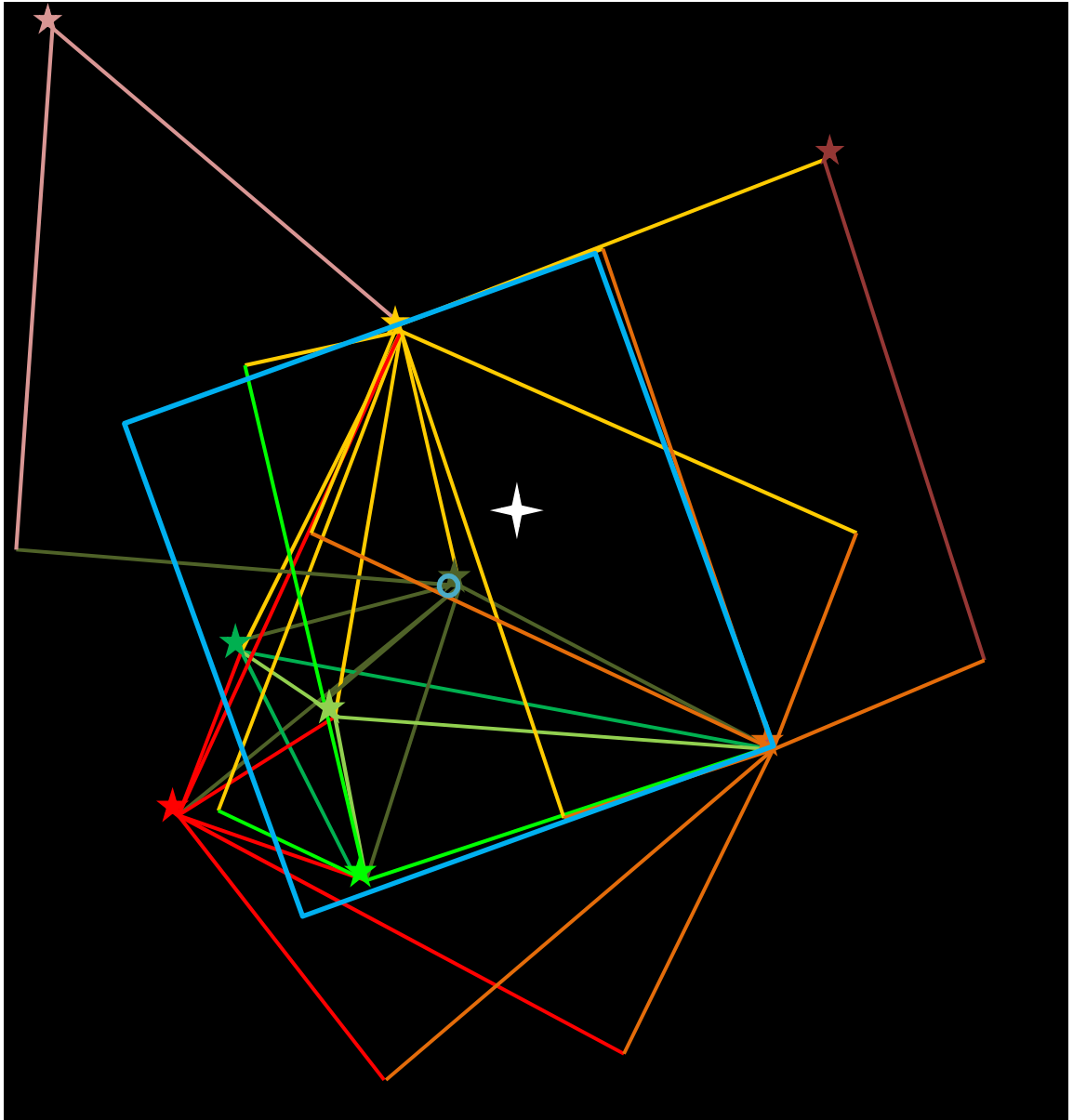


Figure 14-6: a Sketch of the Spiderweb Algorithm in simulation using the same simulated starfield and colour scheme as in Figure 5-6. By drawing the possible orientations of the field such that one target is at the corner and another on the edge of the FoV, it is possible to identify an optimum pointing *and* rotation, shown in blue. This field includes 6 targets as opposed to the 5 shown in Figure 5-6.

In the Spiderweb Algorithm, illustrated in Figure 14-6, the corner of the FoV is placed at one of the potential reference stars. An orientation is then defined such that the edge of the FoV includes another candidate reference star. If the two stars under consideration are less widely separated than the length of the edge of the FoV, then this orientation is defined by the line between those points – the edge of the FoV lies along that line.

If the two candidates are more widely separated than the length of the edge of the FoV, but less widely separated than the diagonal, then it is still possible to align the edges of the FoV between the two targets, but the two stars will be on different edges, with a corner in between.

Assuming a square FoV, and as one of the candidates is at one corner of the FoV, the position of the corner between the two stars may be defined as the vertex opposite to the hypotenuse of a right-angled triangle, where that hypotenuse is defined by the positions of the two candidate reference stars, and where one of the edges is defined as the length of the edge of the FoV.

In Cartesian coordinates, given two candidate reference stars at (x_1, y_1) and (x_2, y_2) , the position of that corner (x_c, y_c) may be calculated given the length of the side of the FoV, R as follows

- First, the length of the hypotenuse between (x_1, y_1) and (x_2, y_2) may be calculated from the standard distance between two points formula.

$$L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- From Pythagoras, the length of the other side, r , may then be calculated.

$$a^2 + b^2 = c^2 \Rightarrow R^2 + r^2 = L^2 \Rightarrow L^2 - R^2 = r^2$$

- The position of the corner (x_c, y_c) can therefore be defined any point r away from one candidate and R away from the other.
- This can be calculated at either of the points of intersection between two circles with formulae $(x - x_1)^2 + (y - y_1)^2 = R^2$ and $(x - x_2)^2 + (y - y_2)^2 = r^2$
- Peterson gives the following solution to the simultaneous equations of two circles. [188]

$$\text{Let } (x_2 - x_1) = x_{diff} \text{ and } (y_2 - y_1) = y_{diff}$$

$$\text{Let } k = \frac{(L^2 + R^2 - r^2)}{2L},$$

Where k is defined as the separation between (x_1, y_1) and the line joining the points of intersection.

$$x = x_1 + \left(\frac{x_{diff}k}{L}\right) + \left(\frac{y_{diff}}{L}\sqrt{R^2 - k^2}\right) \text{ and } y = y_1 + \left(\frac{y_{diff}k}{L}\right) - \left(\frac{x_{diff}}{L}\sqrt{R^2 - k^2}\right)$$

or

$$x = x_1 + \left(\frac{x_{diff}k}{L}\right) - \left(\frac{y_{diff}}{L}\sqrt{R^2 - k^2}\right) \text{ and } y = y_1 + \left(\frac{y_{diff}k}{L}\right) + \left(\frac{x_{diff}}{L}\sqrt{R^2 - k^2}\right)$$

Equation 14-2: Definition of the position of a corner of the FoV in the Spiderweb Algorithm. This position is either of the positions of the points of intersection between two circles, centred on two candidate reference stars, with the radius of one being the length of the edge of the FoV. The two radii form the sides of a right-angled triangle with the distance between the two centres as the hypotenuse .

If the two candidates are more widely separated than the length of the diagonal of the FoV (e.g. $\sqrt{2}$ times the FoV edge size for a square field), they cannot be included in an FoV together, and no alignment is possible.

While the theoretical basis for the Spiderweb Algorithm has been shown to be sound, a full implementation of the Spiderweb Algorithm has not yet been implemented. In summary, the following are considered the key changes required for implementation of this new algorithm

- Transformation of the mathematical solution given in Equation 14-2 to the Polar Coordinate system in which stellar positions are recorded. It is likely that this translation would not be able to use the correction factor used in the Locus Algorithm due to the complexity of the equation
- Definition of algorithmic steps to identify stars which are in an FoV defined by the Spiderweb Algorithm
- Design, development and implementation of a software module to replace the Locus Algorithm module used in this project. Most other software modules would remain unchanged.
- Evaluation of the impact on processing time of the evaluation of both positional and rotational parameters. In addition to increasing the constant time per

calculation due to greater complexity, it is predicted that adding an additional factor to the central algorithm would provide an $O(n)$ multiplier to the calculation of pointing from among n candidate reference stars.

- Comparison of the results of this evaluation with available computational resources to determine whether a complete implementation of this revised algorithm is possible, or whether some further computational optimisations must be sought first.

14.3.2. Scoring

The scoring method used in this project provides a means by which pointings can be compared by providing each candidate reference star with a rating, and adding the ratings for those candidates together. The rating is based on how similar the colour of the reference is to that of the target using two colour indices ($g-r$ and $r-i$ in the case of the catalogues developed in this project).

Refinements to this scoring system are possible due to the modular nature of the software solution. Improvements may be implemented by replacing one or more modules in the Locus Algorithm program with new modules as described below.

Subsection 14.3.2.1 discusses a mechanism by which it is proposed to evaluate the scoring mechanism experimentally, which may be used to test the effects of the proposals in this Chapter.

Subsection 14.3.2.2 discusses some methods by which it may be possible to achieve a better scoring system by incorporating the contribution of additional parameters to the calculation of score.

Finally, Subsection 14.3.2.3 discusses a proposal to add a correction factor to the calculation of score, to account for the possibility that additional reference stars do not linearly contribute to photometric precision.

14.3.2.1 Experimental Evaluation

At time of writing, experiments are being carried out in association with CIT and Raheny Observatory to extend upon the observations discussed in Subsection 10.4. It is

intended that these experiments would permit a more precise calibration for the scoring system.

The experiments consist of on-going observations of a number of fields with targets that have been processed as part of the Exoplanet Catalogue, and assigned pointings and high scores. It is expected to be possible to use observations of high scoring fields to calibrate the scoring system for both high and low scores – the low scores may be simulated by deliberately excluding reference stars from processing.

By using a set of images taken at one particular time using the same device, other contributory factors to photometric precision such as sky brightness can be held constant. This will permit a comparison to be drawn between the number and predicted quality of reference stars, and the observed photometric uncertainty.

In addition, the Algorithm as used in this project excluded many more star than were considered potential reference stars. Observational data can thus be compared with the effects of changing input and other parameters as discussed in Section 14.1.

14.3.2.2 Additional Variables

The scoring system at present assigns a score to a target based on how well the target and the references are matched in two colour indices (e.g. $g-r$ and $r-i$.) It may prove valuable to include additional parameters when calculating score to achieve higher precision.

The current Local Catalogue format includes astrometric (RA, Dec) and photometric (u , g , r , i , and z) data from SDSS, but excludes much of the additional information present in the catalogue. Within this data, three additional features exist which may be added to the equation of rating.

- **Position:** A central tenet of differential photometry is that the atmosphere affects stars that are close to one another in a similar manner. Burdanov et al. suggest that the further apart the target and the reference, the worse this atmospheric correlation would be, which becomes significant at separations greater than 7 arcminutes. [77] It may be beneficial to include a term to account for the separation of target and reference into the scoring mechanism.

- **Magnitude:** According to Milone and Pel, an ideal reference star is similar to or slightly brighter than the target. [1] The current algorithm accepts all stars within a specified magnitude range of the target, and rejects any outside that range, but makes no distinction within that range. If a calibration curve can be identified, it may be better to weight the rating for references in favour of those slightly brighter than the target.
- **Colour:** At present, the algorithm incorporates two colour indices to the scoring mechanism, and thus accounts for three of the five SDSS magnitude bands. Incorporating the other available colour indices may provide additional precision in identifying the degree of similarity between the target and the references. It may be necessary to weight the emphasis placed on colour indices based on the magnitude band to be used in a particular observing run.

In addition, it has been proposed to extend the data incorporated in the Local Catalogue. This extension would allow for more information to be used in calculating the score. The following suggestions have been put forth for additional parameters that may be valuable

- **Known variability:** A variable reference star introduces a systematic error in any observations. Any star which is variable on the timescale of the phenomena to be observed should therefore be excluded, if possible. [189] If this information can be incorporated into the generation of the Local Catalogue, and known variables removed by the Filter step the resulting photometry could be improved
- **Full spectral match:** A project is underway at ITTD to investigate the relationship between broadband magnitudes in the SDSS colour space and the full spectra available as part of the SDSS spectroscopic surveys. As part of this new project, it is intended to investigate the relationship between spectrum and ratings using the Locus Algorithm. It may be possible to build upon this to provide fine-grained calibration of the ratings for particular references.
- **Metallicity:** According to Fan, the SDSS colour distribution shown in Figure 12-5 can be attributed in part to the metallicity of the stars in the catalogue. [170] According to Buchhave et al., the formation of larger planets requires

high metallicity. [190] In order to facilitate the search for these planets, adjusting the score based on metallicity may be advisable.

14.3.2.3 Diminishing Returns

At present, the scoring system works by adding together the ratings of all reference stars in the final FoV to provide a score for that FoV. This system, therefore, makes the assumption that adding more reference stars will always be equally beneficial, and that, all other factors being equal, results should improve in direct proportion to the number of reference stars.

It may be that this assumption is naïve – for example, while it can be predicted that 10 reference stars of equal value would permit better observations than 5, it may be incorrect to assign a score twice as high. It is therefore necessary to consider adding a correction factor to reduce the contribution of later references to the score for a given ensemble.

14.3.3. Computational Optimisation

During the design phase of this project, a number of steps were taken to optimise computational performance, however the emphasis has been on reliable functionality, not maximising efficiency. As a result, there are areas in which there is scope for improvement. Any future project which performs full-scale operation of the project should incorporate a more thorough monitoring of overall performance to identify any factors which do not become apparent in unit testing, as discussed in Section 13.3

Two areas are specified below – Processing efficiency and Data I/O times. In deciding which functions should be improved upon, it is important to identify whether that improvement would significantly affect performance. As was shown shown in Table 13-2, the dominant portion of the time for a job varies depending on the nature of the job – for example, if file I/O is dominant, as it was for the Quasar catalogue (157:1), then increases in processing efficiency will provide only marginal benefits. On the other hand, for the Exoplanet catalogue the balance between data and processing was more even (1.12:1), and as a result, improvements to either aspect would be expected to provide marked improvements in performance.

14.3.3.1 Data I/O

Data I/O contributes a significant proportion of the overall job processing time. For target-by-target processing as was used in the quasar catalogue, it dominates by a factor of 156 as shown in Table 13-2 and discussed in Section 13.3. As a result it is important to reduce this element of job processing. Data I/O can be considered under the following three headings, listed in order of priority.

- **Grid Access:** Downloading files from the LFC takes about 2 second as discussed in Subsection 3.3.2.3 – access authentication dominates hugely compared with network I/O. A temporary solution was implemented during the latter stages of the exoplanet catalogue run, where much of the contents of the Local Catalogue were transferred to a Network File System (NFS) to which the nodes had access. This permitted faster access to data and prevented the LFC from being overwhelmed with I/O requests. Future implementations of this project are likely to use cloud computing as discussed in Section 14.2. This will eliminate this issue which is specific to grid computing. It is unknown at this time whether comparable issues may arise when using equivalent cloud solutions, such as the S3 component of the AWS.
- **Disk I/O:** In this project, files copied from the LFC are accessed locally by the grid nodes. Fields are grouped together to produce jobs. For 0.25 degree FoV, each field has approximately 20 neighbours which need to be included with it. Conversely, therefore, each file must be accessed an average of 21 times (once for its own field, and once for each of its neighbours) to create a mosaic as discussed in Subsection 5.2.1. This creates a potential for improvement by a factor of up to 21 in disk I/O if this inefficiency can be addressed by optimising with an “open once” file I/O strategy.
- **Memory Allocation:** The repeated iterations of the Locus Algorithm make this project particularly vulnerable to memory leaks. In addition, there are no explicit limits set on the size of the input, intermediate or output arrays in the program. This means that excessively large grid jobs have the potential to exceed available resources. While the former issue can be prevented by proper

programming techniques, the latter requires a thorough assessment of the resources consumed by the program while operating.

14.3.3.2 Processing

Processing optimisations take the form of minimising the time taken per star to calculate optimum pointings. As discussed in Chapter 13, the time taken to process a single star is relatively short; however the generation of catalogues requires that this process be iterated many times. Three areas in which there is potential for optimisation have been identified

- **Searching:** The process by which a score is assigned to a field is currently an $O(n^3)$ process over the number of potential reference stars – Potential pointings are identified by calculating the intersection between the boundary boxes for each reference with each other reference, then observing whether each other reference is within a FoV of that point. Optimisation of this segment, at the core of the project, may take the form of a pre-processing step which might potentially reduce unnecessary iterations of the search process.
- **Grouping:** In the Catalogue Traversal mode, the software groups targets by field, to reduce redundant file read calls. Further savings in redundancies may be possible by grouping targets in different sized batches, or by increasing the number of functions carried out at the batch level of the process.
- **Calculation:** the arithmetic functions used in this project are relatively simple, but the repeated iterations can lead to long processing times. It may be possible to further improve the efficiency by optimising these calculations and avoiding repetitions. In addition, some of the proposed modifications to the software, especially the inclusion of rotational components as suggested in Subsection 14.3.1.3, would increase the computational complexity significantly. Should these proposals be followed, it is imperative that a careful optimisation strategy be implemented to minimise the complexity of the calculation, especially by eliminating any unnecessary trigonometric calculations.

14.4. Conclusions

Proposed future Projects			
Reuse of existing project systems			
Proposed work	Scope of work	Deliverables	Expected use
Sample analysis of project outputs using different input parameters	Use of stratified sample of existing pointings for different combinations of factors	Metrics on impact of source and user input parameters on score	Contribution to refinement of scoring system (see below)
Generation of new catalogue using different input parameters	Full scale run of catalogue generation pipeline for different input or observational parameters using suitable HPC solution	Catalogue comparable to the existing Exoplanet catalogue for further bands or observatory conditions	Enabling the use of the catalogue with telescopes under conditions other than those for which the existing catalogue was developed.
Incorporation of additional data sources to the data analysis pipeline	Modification of SDSS-specific software modules to allow them to incorporate other data sources	System capable of analysing data from multiple source systems	Expansion of the data analysis pipeline to incorporate data from new catalogues such as LSST or existing all-sky catalogues such as USNO
Migration to Cloud			
Proposed work	Scope of work	Deliverables	Expected use
Migration of grid-based data storage and pipelines to cloud-based systems	Modification of grid modules and replacement with suitable cloud-based systems (e.g. AWS)	Cloud-based storage and analysis system which enables future use of the data analysis pipeline.	Enabling generation of new catalogues and analyses of the output data.
Refinements of existing systems			
Proposed work	Scope of work	Deliverables	Expected use
Addition of functionality to data analysis systems	Accommodation of exceptions which arose from the design of the software, and removal of limits such as near-polar fields and allowance for rotating FoV	System capable of adapting to additional user requirements for <i>ad hoc</i> queries or catalogue generation	Expanded capability of existing systems with regards to areas of the celestial sphere which were previously unsuitable, capacity to permit the user to rotate their telescope to further optimise their pointings
Refinement of Scoring system	Calibration of scoring system by comparing observed and simulated lightcurves with existing scoring systems	More robust scoring system which provides improved precision in determination of optimum pointings for different requirements	Enables more sophisticated analysis using modular scoring systems for different roles if needed
Computational optimisation	Algorithmic analysis of the implemented software systems to enable optimisation to performance	Refined software system with faster computational performance and runtime	Enable more iterations of the software system with given computational resource

Table 14-1: Summary of proposed expansions to this project

This project has demonstrated a system for analysing large astronomical catalogues to produce ranked output catalogues of pointings and scores given particular input parameters. To further develop that system, it will be necessary to implement a new HPC model. AWS offers an example of a cloud computing model conceptually similar to Grid Ireland to which this project might be migrated with minimal modifications.

With a suitable system in place, it is proposed to repeat the use of the software generated for this project to generate new catalogues and pointings and explore the parameter space of the system. This exploration must account for photometric parameters of target and reference stars, observational parameters such as field of view size, and input data parameters such as source catalogues.

It may also be possible, with a new computing paradigm, to explore potentially valuable refinements to the software which were beyond the scope of this project. Additional functions which were not included in the initial release of the software may be valuable, including making allowance for areas of the sky excluded from the current edition of the software. Addition of functions such as camera rotation must be balanced against the additional computational demand imposed.

The original scoring system used in this project requires more rigorous calibration than was possible within the scope of this project. This calibration will require experimental support, and may be developed, with appropriate software modifications, to incorporate such refinements as additional variables or a “law of diminishing returns” for densely-populated fields.

Finally, the software component of the project should itself be subject to review for efficiency and performance issues. The choice of emphasis between data-based and processing-based performance issues is partly dependant on which component dominates the overall job completion time in the mode of operation to be optimised. Processor-based efficiencies depend on optimising search and grouping strategies, and upon minimising processor-intensive operations such as trigonometry. Data-oriented strategies depend in part upon the HPC strategy, upon careful planning of grid jobs to ensure files are not accessed unnecessarily, and upon control of memory allocation.

15. Conclusions

This project developed a technique, known as the Locus Algorithm, to produce optimised pointings for ensemble differential photometry for a given target, given information from a catalogue on the stars around it. This technique was used in a software and hardware system which repeated this analysis on large data sets using grid computing.

Two output catalogues were generated, the Quasar Catalogue and the Exoplanet Catalogue. The Quasar Catalogue analysed 77,429 quasars and produced optimised pointings for 23,697 of those targets. The Exoplanet Catalogue analysed ~86,000,000 stars from SDSS and produced optimised pointings for 61,662,376 of them. Statistically, it is likely that many of these stars play host to exoplanets

These two catalogues represent a resource that can guide observations on any of the targets for which there are pointings. Further, the scores provided can suggest a preference, for an observer who has a choice of targets but no other *a priori* preference between them. The information in the Exoplanet Catalogue, for example, can guide an otherwise “blind” search for exoplanets by indicating targets for which higher photometric precision may be available due to the better set of reference stars.

Detailed meta-analysis of the Exoplanet Catalogue allowed for clearer interpretation of the score for a target. Scores showed substantial variation. The mean score in the Exoplanet Catalogue overall was 6.72, and therefore any target with a score higher than this can be considered “above average.” However, the distribution of scores showed a long tail. The star with the highest score that was analysed was SDSS J203733.62+001953.5, with a score of 117.7. This indicated that a user looking to select a field with a large number of very similar references can search for those with serendipitously high scores. These fields may prove particularly useful for survey work, as a single telescope can be used to monitor many very similar stars.

In addition, the meta-analysis set boundaries on the reliability of the output from the Exoplanet Catalogue. It is not advised to use the pointings for stars which do not fall in the r magnitude range $16 < r < 20.2$ and use of stars with colour outside $0.27 < g-r < 1.86$

and $0.05 < r-i < 1.63$ is discouraged. Targets with $r \approx 17.5$ and $(g-r) \approx 0.450$ or 1.4 and $(r-i) \approx 0.150$ show the best results.

Assessment of the computing solution provided some insight into the operation of grid computing on the “Data Rich, Process Poor” nature of this project. By using unit testing to guide grid job submission, it became clear that the Data Access component of the project dominated over the processing aspect in the API and Quasar pipeline jobs by a factor of 156:1 and 157:1 respectively.

Unit testing on the Exoplanet Catalogue indicated that it would not be as strongly affected by this issue. Scaling up to full-scale grid deployment of the Exoplanet pipeline led to a different conclusion, however. During the grid job, the GMS was unable to handle the 19.5 requests per second that the grid jobs were submitting at their peak. As a result, Data Access actually became a blocking issue. It was necessary to implement an alternative Data Access method, by making the data available on a NFS connected via LAN to the WNs. This provided much faster data access which led to processing and grid management overhead becoming the primary contributors to grid run time.

Grid jobs were observed to have a reliability on par with or better than the 80% typically expected of user grid jobs [179], in the exoplanet ($^{1598}/_{1791} = 89.22\%$) and API ($^{359}/_{422} = 85.1\%$) jobs, but fell down in the quasar job ($^{40}/_{78} = 51\%$). It is thought that loss of data on the quasar job may have occurred outside the grid jobs, which would bring down this result.

The Locus Algorithm system has been demonstrated to operate in all of its designed modes. It is proposed that this system be refined and reused. Migration to Cloud Computing will provide additional challenges, but will provide up-to-date technology with which to perform repeat analysis of the catalogue generation jobs. The phase space of the various parameters used to define the catalogues that are produced, such as target lists, FoV size, colour and magnitude limits, and bandpass used can be explored in depth. This will allow the iterative refinement of the scoring system by experimental assessment of the validity of results. Such refinement of the scoring system may also incorporate additional data into the catalogue.

Finally, the availability of a cloud computing solution will allow the project to be tested in different virtual server environments, which will help to guide changes to the code that will optimise the computing performance

..

Appendix A Bibliography & References

A-a In-Text citations

- [1] E. F. Milone and J. W. Pel, “The High Road to Astronomical Photometric Precision: Differential Photometry,” in *Astronomical Photometry*, New York, Springer, 2011, pp. 38-68.
- [2] S. O'Driscoll, *private communication*, Cork, 2007.
- [3] O. Creaner, E. Hickey, K. Nolan, T. O Briain and N. Smith, “Large-Catalogue Optimisation of Quasar Differential Photometry Fields by Grid Computing,” in *Astronomical Data Analysis Software and Systems XIX*, 2010.
- [4] S. Chakrabarti, M. Ester, U. Fayyad, J. Gehrke, J. Han, S. Morishita, G. Piatetsky-Shapiro and W. Wang, “Data Mining Curriculum: A Proposal (Version 1.0),” Intensive Working Group of ACM SIGKDD Curriculum Committee, 2006.
- [5] R. Gal and S. Jester, “SDSS Sky Coverage notes for DR7,” Xeno Media, 1 August 2013. [Online]. Available: <http://www.sdss2.org/dr7/>. [Accessed 6 August 2014].
- [6] Sloan Digital Sky Survey Team, “SDSS Query/CASJobs,” Alfred P. Sloan Foundation, 11 February 2013. [Online]. Available: <http://casjobs.sdss.org/CasJobs/>. [Accessed 14 February 2013].
- [7] S. Sciaba, S. Burke, E. Campana, M. Lanciotti, P. Litmaath, V. Lorenzo, C. Miccio, R. Nater and Santinelli, GLite 3.2 User Guide, CERN, 2011.
- [8] A. H. Batten, “Two centuries of study of Algol systems,” *Space Science Reviews*, vol. 49, no. 3, pp. 1-8, 1989.
- [9] I. Zolotukhin, F. Roques, J. Schneider, C. Chauvin, M. Mancini, P. Le Sidaner, A. Sergeev, M. Chernyshov, R. Savalle, J. Normand and C. Dedieu, “The Extrasolar Planets Encyclopaedia,” Observatoire de Paris, 24 January 2017. [Online]. Available:

<http://exoplanet.eu/>. [Accessed 25 January 2017].

- [10] A. Giltinan, D. Loughnan, A. Collins and N. Smith, “Using EMCCD's to improve the photometric precision of ground-based astronomical observations,” in *Journal of Physics: Conference Series*, 2011.
- [11] R. L. Gilliland and T. M. Brown, “Time-resolved CCD photometry of an ensemble of stars,” *Publications of the Astronomical Society of the Pacific*, vol. 100, pp. 754-765, 1988.
- [12] G. H. A. Cole, *Wandering Stars: About Planets and Exo-Planets: An Introductory Notebook*, Hull: World Scientific, 2006.
- [13] E. Budding and O. Demircan, *Introduction to Astronomical Photometry*, 2nd ed., Cambridge: Cambridge University Press, 2007.
- [14] I. Robson, *Active Galactic Nuclei*, Chichester: J. Wiley & sons, 1996.
- [15] N. N. Samus, O. V. Durlevich, E. Kazarovets, N. Kireeva, E. N. Pastukhova, A. V. Zharova and e. al, “General Catalogue of Variable Stars, VizieR On-line Data Catalog: B/gcvs,” Moscow, 2007-2012.
- [16] J. Percy, *Understanding Variable Stars*, Cambridge: Cambridge University Press, 2007.
- [17] M. Perryman, *The Exoplanet Handbook*, Cambridge: Cambridge University Press, 2011.
- [18] A. Wolszczan and F. Dale, “A planetary system around the millisecond pulsar PSR1257 + 12,” *Nature*, vol. 355, no. 6356, pp. 145-147, 9 January 1992.
- [19] M. Mayor and D. Queloz, “A Jupiter-mass companion to a solar-type star,” *Nature*, vol. 378, no. 6555, p. 355–359, 1995.
- [20] D. Charbonneau, T. M. Brown, D. W. Latham and M. Mayor, “Detection of Planetary Transits across a Sun-like Star,” *The Astrophysical Journal*, vol. 529, no. 1, pp. L45-L48, 2000.

- [21] G. W. Henry, G. W. Marcy, R. P. Butler and S. S. Vogt, “A Transiting “51-Peg-Like” Planet, , 529:L41,” *The Astrophysical Journal*, vol. 529, no. 1, pp. L41-L44, 2000.
- [22] N. Smith, C. Coates, A. Giltinan, J. Howard, A. O'Connor, S. O'Driscoll, M. Hauser and S. Wagner, “EMCCD technology and its impact on rapid low-light photometry,” in *Optical and Infrared Detectors for Astronomy*, 2004.
- [23] N. Smith, 2007-2014.
- [24] V. Trimble and M. J. Aschwanden, “Astrophysics in 2000,” *Publications of the Astronomical Society of the Pacific*, vol. 113, no. 787, pp. 1025-1114, 2001.
- [25] J. Lean, J. Beer and R. Bradley, “Reconstruction of solar irradiance since 1610: Implications for climate change,” *Geophysical Research Letters*, vol. 22, no. 23, pp. 3195-3198, 1995.
- [26] D. Hoffleit, “History of the Discovery of Mira Stars,” *Journal of the American Association of Variable Star Observers*, vol. 25, no. 2, pp. 115-136, 1996.
- [27] S. R. Wilk, “Mythological Evidence for Ancient Observations of Variable Stars,” *The Journal of the American Association of Variable Star Observers*, vol. 24, no. 2, pp. 129-133, 1996.
- [28] J. Goodricke, “A Series of Observations on, and a Discovery of, the Period of the Variation of the Light of the Bright Star in the Head of Medusa, Called Algol. in a Letter from John Goodricke, Esq. to the Rev. Anthony Shepherd, DDFRS and Plumian Professor at Cambridge,” *Philosophical Transactions of the Royal Society of London*, vol. 73, pp. 474-482, 1783.
- [29] A. S. Meltzer, “A Spectroscopic Investigation of Algol,” *Astrophysical Journal*, vol. 125, p. 359, 1957.
- [30] D. M. Kipping, “Transit timing effects due to an exomoon,” *Monthly Notices of the Royal Astronomical Society*, vol. 392, no. 3, pp. 181-189, 2009.

- [31] I. D. Howarth, "On stellar limb darkening and exoplanetary transits," *Monthly Notices of the Royal Astronomical Society*, vol. 418, no. 2, pp. 1165-1175, 2011.
- [32] Working Group on Extrasolar Planets (WGESP) of the International Astronomical Union, "Position Statement on the Definition of a "Planet"," 28 February 2003. [Online]. Available: <http://astro.berkeley.edu/~basri/defineplanet/IAU-WGExSP.htm>. [Accessed 8 September 2014].
- [33] T. P. Castellano, G. Laughlin, R. S. Terry, M. Kaufman, S. Hubbert, G. M. Schelbert, D. Bohler and R. Rhodes, "Detection of Transits of Extrasolar Giant Planets With Inexpensive Telescopes and CCDs," *The Journal of the American Association of Variable Star Observers*, vol. 33, no. 1, pp. 1-24, 2004.
- [34] P. Sherrer, "NASA's SDO Satellite Captures 2012 Venus Transit," NASA/SDO, HMI, 5 June 2012. [Online]. Available: <https://www.flickr.com/photos/gsfcr/7158100537/in/photostream/>. [Accessed 9 September 2014].
- [35] L. Billings, "Exoplanets on the cheap (London) 470 (2011): 27-29.," *Nature*, vol. 470, no. 7332, pp. 27-29, 2011.
- [36] A. Cox, Ed., *Allen's Astrophysical Quantities*, 4th ed., New York: Springer, 2000.
- [37] D. Koch and A. Gould, "Characteristics of Transits," NASA/Kepler, 30 June 2005. [Online]. Available: http://certificate.ucl.ac.uk/modules/year_one/NASA_Kepler/character.html. [Accessed 9 September 2014].
- [38] G. Chabrier, I. Baraffe, J. Leconte, J. Gallardo and T. Barman, "The mass-radius relationship from solar-type," *AIP Conference Proceedings*, vol. 1094, no. 1, pp. 102-111, 2009.
- [39] European Southern Observatory, "VLTI observations of the radii of four small stars," 29 November 2002. [Online]. Available:

<https://www.eso.org/public/australia/images/eso0232c/>. [Accessed 25 January 2017].

- [40] V. Springel, S. D. M. White, A. Jenkins, C. S. Frenk, N. Yoshida, L. Gao, J. Navarro, R. Thacker, D. Croton, J. Helly, J. A. Peacock, S. Cole, P. Thomas, H. Couchman, A. Evrard, J. Colberg and F. Pearce, “Simulations of the formation, evolution and clustering of galaxies and quasars,” *Nature*, vol. 435, no. 7042, pp. 629-636, 2005.
- [41] N. Smith, A. Giltinan, A. O'Connor, S. O'Driscoll, A. Collins, D. Loughnan and A. Papageorgiou, “EMCCD Technology in High Precision Photometry on Short Timescales,” in *High Time Resolution Astrophysics*, D. Phelan, O. Ryan and A. Shearer, Eds., Berlin, Springer, 2008, pp. 257-279.
- [42] M.-P. Véron-Cetty and P. Véron, “A catalogue of quasars and active nuclei,” *Astronomy and Astrophysics*, vol. 518, no. 10, 2010.
- [43] K. N. Abazajian, J. K. Adelman-McCarthy, M. A. Agüeros, S. S. Allam, C. Allende Prieto, D. An, K. S. J. Anderson, S. F. Anderson, J. Annis, N. A. Bahcall, C. A. L. Bailer-Jones, J. C. Barentine, B. A. Bassett, A. C. Becker, T. C. Beers, E. F. Bell, V. Belokurov, A. A. Berlind, E. F. Berman, M. Bernardi, S. J. Bickerton, D. Bizyaev, J. P. Blakeslee, M. R. Blanton, Bochanski, John J., Boroski, William N., Brewington, Howard J., Brinchmann, Jarle, Brinkmann, J., Brunner, Robert J., Budavári, Tamás, Carey, Larry N., Carliles, Samuel, Carr, Michael A., Castander, Francisco J., Cinabro, David, Connolly, A. J., Csabai, István, Cunha, Carlos E., Czarapata, Paul C., Davenport, James R. A., de Haas, Ernst, Dilday, Ben, Doi, Mamoru, Eisenstein, Daniel J., Evans, Michael L., Evans, N. W., Fan, Xiaohui, Friedman, Scott D., Frieman, Joshua A., Fukugita, Masataka, Gänsicke, Boris T., Gates, Evalyn, Gillespie, Bruce, Gilmore, G., Gonzalez, Belinda, Gonzalez, Carlos F., Grebel, Eva K., Gunn, James E., Györy, Zsuzsanna, Hall, Patrick B., Harding, Paul, Harris, Frederick H., Harvanek, Michael, Hawley, Suzanne L., Hayes, Jeffrey J. E., Heckman, Timothy M., Hendry, John S., Hennessy, Gregory S., Hindsley, Robert B., Hoblitt, J., Hogan, Craig J., Hogg, David W., Holtzman, Jon A., Hyde, Joseph B., Ichikawa, Shin-ichi, Ichikawa, Takashi, Im, Myungshin, Ivezić, Željko, Jester, Sebastian, Jiang, Linhua, Johnson, Jennifer A., Jorgensen, Anders M., Jurić, Mario, Kent, Stephen M., Kessler, R., Kleinman, S. J.,

Knapp, G. R., Konishi, Kohki, Kron, Richard G., Krzesinski, Jurek, Kuropatkin, Nikolay, Lampeitl, Hubert, Lebedeva, Svetlana, Lee, Myung Gyoong, Lee, Young Sun, French Leger, R., Li, Nolan, Lima, Marcos, Lima, Marcos, Lin, Huan, Long, Daniel C., Loomis, Craig P., Loveday, Jon, Lupton, Robert H., Magnier, Eugene, Malanushenko, Olena, Malanushenko, Viktor, Mandelbaum, Rachel, Margon, Bruce, Marriner, John P., Martínez-Delgado, David, Matsubara, Takahiko, McGehee, Peregrine M., McKay, Timothy A., Meiksin, Avery, Morrison, Heather L., Mullally, Fergal, Munn, Jeffrey A., Murphy, Tara, Nash, Thomas, Nebot, Ada, Neilsen, Eric H., Jr., Newberg, Heidi Jo, Newman, Peter R., Nichol, Robert C., Nicinski, Tom, Nieto-Santisteban, Maria, Nitta, Atsuko, Okamura, Sadanori, Oravetz, Daniel J., Ostriker, Jeremiah P., Owen, Russell, Padmanabhan, Nikhil, Pan, Kaike, Park, Changbom, Pauls, George, Peoples, John, Jr., Percival, Will J., Pier, Jeffrey R., Pope, Adrian C., Pourbaix, Dimitri, Price, Paul A., Purger, Norbert, Quinn, Thomas, Raddick, M. Jordan, Re Fiorentin, Paola, Richards, Gordon T., Richmond, Michael W., Riess, Adam G., Rix, Hans-Walter, Rockosi, Constance M., Sako, Masao, Schlegel, David J., Schneider, Donald P., Scholz, Ralf-Dieter, Schreiber, Matthias R., Schwobe, Axel D., Seljak, Uroš, Sesar, Branimir, Sheldon, Erin, Shimasaku, Kazu, Sibley, Valena C., Simmons, A. E., Sivarani, Thirupathi, Allyn Smith, J., Smith, Martin C., Smolčić, Vernesa, Snedden, Stephanie A., Stebbins, Albert, Steinmetz, Matthias, Stoughton, Chris, Strauss, Michael A., SubbaRao, Mark, Suto, Yasushi, Szalay, Alexander S., Szapudi, István, Szkody, Paula, Tanaka, Masayuki, Tegmark, Max, Teodoro, Luis F. A., Thakar, Aniruddha R., Tremonti, Christy A., Tucker, Douglas L., Uomoto, Alan, Vanden Berk, Daniel E., Vandenberg, Jan, Vidrih, S., Vogeley, Michael S., Voges, Wolfgang, Vogt, Nicole P., Wadadekar, Yogesh, Watters, Shannon, Weinberg, David H., West, Andrew A., White, Simon D. M., Wilhite, Brian C., Wonders, Alainna C., Yanny, Brian, Yocum, D. R., York, Donald G., Zehavi, Idit, Zibetti, Stefano and Zucker, Daniel B., “The Seventh Data Release of the Sloan Digital Sky Survey,” *The Astrophysical Journal Supplement*, vol. 182, no. 2, pp. 543-558, 2009.

- [44] A. U. Landolt, “Johnson Photometry and Its Descendants,” in *Astronomical Photometry: Past, Present and Future*, E. F. Milone and C. Sterken, Eds., New York, Springer, 2011, pp. 109-125.

- [45] A. U. Landolt, “UBVRI photometric standard stars around the celestial equator,” *The Astronomical Journal*, vol. 88, pp. 439-460, 1983.
- [46] C. Sterken and J. Manfroid, *Astronomical Photometry: A Guide*, New York: Springer, 1992.
- [47] C. W. Engelke, S. D. Price and K. E. Kraemer, “Spectral Irradiance Calibration in the Infrared. XVII. Zero-magnitude Broadband Flux Reference for Visible-to-infrared Photometry,” *The Astronomical Journal*, vol. 140, no. 6, pp. 1919-1928, 2010.
- [48] M. Cohen, “Absolute Photometry: Past and Present,” in *Astronomical Photometry: Past, Present and Future*, New York, Springer, 2011, pp. 179-188.
- [49] H. Kjeldsen and S. Frandsen, “High-precision time-resolved CCD photometry,” *Publications of the Astronomical Society of the Pacific*, vol. 41, no. 3, pp. 413-434, 1992.
- [50] G. Bakos, R. W. Noyes, G. Kovács, K. Z. Stanek, D. D. Sasselov and I. Domsa, “Wide-Field Millimagnitude Photometry with the HAT: A Tool for Extrasolar Planet Detection,” *Publications of the Astronomical Society of the Pacific*, vol. 116, no. 817, pp. 266-277, 2004.
- [51] A. T. Young, R. M. Genet, L. J. Boyd, W. J. Borucki, G. W. Lockwood, G. W. Henry, D. S. Hall, D. P. Smith, S. L. Baliumas, R. Donahue and D. H. Epand, “Precise automatic differential stellar photometry (1991): 221-242.,” *Publications of the Astronomical Society of the Pacific*, vol. 103, no. 2, pp. 221-242, 1991.
- [52] S. B. Howell, A. I. Warnock and K. J. Mitchell, “Statistical error analysis in CCD time-resolved photometry with applications to variable stars and quasars,” *Astronomical Journal*, vol. 95, pp. 247-256, 1988.
- [53] G. H. Herbig, “Stellar Magnitudes 4 (1945): 386.,” *Leaflet of the Astronomical Society of the Pacific*, vol. 4, no. 198, pp. 386-393, 1945.
- [54] Hipparchus, Rhodes?, 127BC.

- [55] C. Ptolemy, *Almagest*, Alexandria, 138.
- [56] C. Sterken, M. E. F. and A. T. Young, “Photometric Precision and Accuracy,” in *Astronomical Photometry: Past, Present and Future*, New York, Springer, 2011, pp. 1-32.
- [57] P. Bouguer, *Traité d'optique sur la gradation de la lumière*, 1760.
- [58] J. B. Hearnshaw, *The measurement of starlight: two centuries of astronomical photometry*, Cambridge: Cambridge University Press, 1996.
- [59] N. Pogson, “Magnitudes of Thirty-six of the Minor Planets for the first day of each month of the year 1857,” *Monthly Notices of the Royal Astronomical Society*, vol. 17, pp. 12-15, 1856.
- [60] G. P. Bond, “Stellar-Photography,” *Astronomische Nachrichten*, vol. 46, no. 6-7, pp. 81-100, 1859.
- [61] I. Elliot, “Monck, William Henry Stanley,” in *Biographical Encyclopedia of Astronomers*, New York, Springer, 2014, pp. 1510-1512.
- [62] G. Minchin, “The Electrical Measurement of Starlight. Observations Made at the Observatory of Daramona House, Co. Westmeath, in April, 1895. Preliminary Report 58.347-352 (1895): 142-154.,” *Proceedings of the Royal Society of London*, vol. 58, pp. 142-154, 1895.
- [63] J. Stebbins, “The measurement of the light of stars with a selenium photometer, with an application to the variations of Algol,” *The Astrophysical Journal*, vol. 32, pp. 185-214, 1910.
- [64] T. Walraven, “On the light-variation of AI Velorum,” *Bulletin of the Astronomical Institutes of the Netherlands*, vol. 11, p. 421, 1952.
- [65] S. B. Howell, “High Precision Differential Photometry with CCDs: A Brief History,” in *Astronomical Photometry: Past, Present and Future*, New York, Springer, 2011, pp. 71-

- [66] S. B. Howell, Handbook of CCD astronomy, vol. 5, Cambridge: Cambridge University Press, 2006.
- [67] A. R. Walker, “CCD observations of photoelectric standard stars,” *Monthly Notices of the Royal Astronomical Society*, vol. 209, pp. 83-91, 1984.
- [68] M. E. Everett and S. B. Howell, “A Technique for Ultrahigh-Precision CCD Photometry,” *Publications of the Astronomical Society of the Pacific*, vol. 133, no. 789, pp. 1428-1435, 2001.
- [69] R. K. Honeycutt, “CCD ensemble photometry on an inhomogeneous set of exposures,” *Publications of the Astronomical Society of the Pacific*, vol. 104, no. 672, pp. 435-440, 1992.
- [70] D. Tody, “The IRAF Data Reduction and Analysis System,” *1986 Astronomy Conferences*, pp. 733-748, October 1986.
- [71] K. Venkataramani, S. Ghetiya, S. Ganesh, U. C. Joshi, V. K. Agnihotri and K. S. Baliyan, “Optical Spectroscopy of Comet C/2014 Q2 (Lovejoy) from MIRO,” *Monthly Notices of the Royal Astronomical Society*, vol. 463, no. 2, pp. 2137-2144, 2016.
- [72] M. D. de La Peña, R. L. White and P. Greenfield, “The PyRAF Graphics System,” *Astronomical Data Analysis Software and Systems X, ASP Conference Proceedings*, vol. 238, pp. 59-62, 2001.
- [73] Space Telescope Science Institute, “PyRAF,” AURA for NASA, January 2017. [Online]. Available: http://www.stsci.edu/institute/software_hardware/pyraf. [Accessed 4 February 2017].
- [74] R. H. Lupton, Z. Ivezic, J. E. Gunn, G. Knapp, M. A. Strauss and N. Yasuda, “SDSS imaging pipelines,” *Astronomical Telescopes and Instrumentation*, pp. 350-356, 2002.
- [75] S. O'Driscoll and N. J. Smith, “The realization of an automated data reduction pipeline

- in IRAF: the PhotMate system,” *Astronomical Telescopes and Instrumentation*, 2004.
- [76] A. W. J. Cousins, “Atmospheric Extinction in Relation to Stellar Photometry,” *Monthly Notes of the Astronomical Society of South Africa*, vol. 44, p. 10, 1985.
- [77] A. Y. Burdanov, V. V. Krushinsky and A. A. Popov, “Astrokit-an efficient program for high-precision differential CCD photometry and search for variable stars,” *Astrophysical Bulletin*, vol. 69, no. 3, pp. 368-376, 2014.
- [78] T. C. Hinse, H. Wonyong, J.-N. Yoon, C.-U. Lee, Y.-g. Kim and C.-H. Kim, “Photometric defocus observations of transiting extrasolar planets,” *Journal of Astronomy and Space Science*, 2015.
- [79] A. Collins and N. Smith, *A new Technique for Improving High Precision Photometry in Turbulent Atmospheric Conditions*, Cork, 2011.
- [80] D. L. Fried, “Probability of getting a lucky short-exposure image through turbulence,” *Journal of the Optical Society of America*, vol. 68, no. 12, pp. 1651-1658, 1978.
- [81] S. Zhang, J. Zhao and J. Wang, “High resolution astronomical image restoration system for large ground-based telescope,” *Chinese Optics Letters*, vol. 10, no. s2, pp. S21004-1-S21004-4, 2012.
- [82] E. F. Milone and A. T. Young, “The Rise and Improvement of Infrared Photometry,” in *Astronomical Photometry: Past, Present and Future*, New York, Springer, 2011, pp. 127-143.
- [83] D. Grennan, *private communication*, Dublin, 2012.
- [84] K. Jordi, E. K. Grebel and K. Ammon, “Empirical color transformations between SDSS photometry and other photometric systems,” *Astronomy and Astrophysics*, vol. 460, no. 1, pp. 339-347, 2006.
- [85] Z. Ivezić, J. A. Tyson, E. Acosta, R. Allsman, S. F. Anderson, J. Andrew and D. Kirkby, “LSST: from science drivers to reference design and anticipated data products,” *arXiv*

preprint, vol. 0805, no. 2366, 2008.

- [86] W. D. Pence, L. Chiappetti, C. G. Page, R. A. Shaw and E. Stobie, "Definition of the Flexible Image Transport System (FITS), version 3.0," *Astronomy and Astrophysics*, vol. 524, no. A42, pp. 1-42, 2010.
- [87] J. Dongarra and A. Lastovetsky, "An overview of heterogeneous high performance and grid computing." (2006): 1-25., in *Engineering the Grid: Status and Perspective*, 2006, pp. 1-25.
- [88] I. Foster, Y. Zhao, I. Raicu and S. Lu, "Cloud computing and grid computing 360-degree compared," *Grid Computing Environments Workshop*, November 2008.
- [89] S. Ghosh, *Distributed systems: an algorithmic approach*, CRC Press, 2010.
- [90] M. Dimitrijević and V. Litovski, "Virtual Machine Technology in Grid Computing," 2008, pp. 240-243.
- [91] A. L. Beberg, D. L. J. G. Ensign, S. Khaliq and V. S. Pande, "Folding@ home: Lessons from eight years of volunteer distributed computing," in *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on Parallel & Distributed Computing*, 2009.
- [92] W. Gropp, E. Lusk and A. Skjellum, "A High-Performance, Portable Implementation of the MPI Message Passing Interface," *Parallel Computing*, vol. 22, no. 6, pp. 789-828, 1996.
- [93] School of Computer Science and Statistics, Trinity College Dublin, "Grid Ireland Opcentre," Trinity College Dublin, 15 June 2009. [Online]. Available: <http://grid.ie/opscentre.html>. [Accessed 6 February 2012].
- [94] B. Coughlan, J. Walsh and D. O'Callaghan, "The Grid-Ireland Deployment Architecture," *European Grid Conference*, pp. 354-363, 2005.
- [95] V. Kunik, "Grid Data Management, Israeli Grid Workshop," Israeli Grid NA3, September 2006. [Online]. Available:

<http://indico.cern.ch/event/a063283/session/s1/contribution/s1t3/material/0/0.ppt>.
[Accessed 2014 September 23].

- [96] C. Panagiotis, “ls man page,” 1994. [Online]. Available: <http://unixhelp.ed.ac.uk/CGI/man-cgi?ls>. [Accessed 23 September 2014].
- [97] B. Gambrel and K. Endsley, Introduction to Programming, C. A. Huchinson, Ed., Que Publishing, 1992.
- [98] D. Bell, I. Morrey and J. Pugh, Software engineering: a programming approach, 2nd ed., Prentice Hall, 1992.
- [99] H. M. Deitel and P. J. Deitel, C How to Program, 3rd ed., Prentice Hall, 2001.
- [100] B. W. Kernighan and D. M. Ritchie, The C Programming Language, 2nd ed., Prentice Hall, 1988.
- [101] H. Schildt, C The Complete Reference, Osborne McGraw-Hill, 1987.
- [102] British Computer Society. Glossary Working Party, The BCS Glossary of ICT and Computing Terms, 11th ed., Pearson: Prentice Hall., 2005.
- [103] A. B. Tucker and R. E. Noonan, Programming Languages: Principles and Paradigms, McGraw Hill, 2007.
- [104] S. M. H. Collin, Dictionary of ICT, 4th ed., Bloomsbury, 2004.
- [105] G. van Rossum, “Python Web Site,” Python Software Foundation, 2017. [Online]. Available: <https://www.python.org>. [Accessed 5 February 2017].
- [106] H. H. Cheng, The Ch Language Environment, 2002.
- [107] P. Tymann and C. Reynolds, Schaum's outline of principles of computer science, McGraw-Hill, Inc., 2008.
- [108] H. M. Deitel and P. J. Deitel, Simply C++ : An Application-Driven Tutorial Approach,

Pearson Prentice Hall, 2005.

- [109] J. Hughes, “Why functional programming matters.,” *The computer journal*, vol. 32, no. 2, pp. 98-107, 1989.
- [110] J. M. Spivey, *An introduction to logic programming through Prolog*, Prentice-Hall, 1995.
- [111] Sloan Digital Sky Survey Team, “SDSS DR7 Home Page,” Alfred P. Sloan Foundation, 31 July 2012. [Online]. Available: <http://www.sdss.org/DR7/>. [Accessed 22 February 2013].
- [112] M. Taylor, “TOPCAT, Tool for OPerations on Catalogues And Tables, Does what you want with tables,” Astrophysics Group, Physics Department, University of Bristol, 23 September 2016. [Online]. Available: <http://www.star.bristol.ac.uk/~mbt/topcat/>. [Accessed 12 February 2017].
- [113] International Virtual Observatory Alliance (IVOA), “IVAO.net,” International Virtual Observatory Alliance (IVOA), December 2016. [Online]. Available: <http://ivoa.net>. [Accessed 12 February 2017].
- [114] I. Ortiz, J. Lusted, P. Dowler, A. Szalay, Y. Shirasak, M. A. Nieto-Santisteban, M. Ohishi, W. O’Mullane, P. Osuna, the VOQL-TEG and the VOQL Working Group, “IVOA Astronomical Data Query Language, Version 2.0,” 30 October 2008. [Online]. Available: <http://www.ivoa.net/documents/REC/ADQL/ADQL-20081030.pdf>. [Accessed 12 February 2017].
- [115] The IVOA Virtual Observatory Query Language (VOQL) working group members, The IVOA Data Access Layer (DAL) working group members, “Astronomical Data Query Language, Version 2.1,” 2 May 2016. [Online]. Available: <http://www.ivoa.net/documents/ADQL/20160502/WD-ADQL-2.1-20160502.pdf>. [Accessed 12 February 2017].
- [116] F. Ochsenbein, R. Williams, C. Davenhall, M. Demleitner, D. Durand, P. Fernique, D. Giarretta, R. Hanisch, T. McGlynn, A. Szalay, M. Taylor and A. Wicenec, “VOTable

- Format Definition Version 1.3,” 20 September 2013. [Online]. Available: <http://www.ivoa.net/documents/VOTable/20130920/REC-VOTable-1.3-20130920.pdf>. [Accessed 12 February 2017].
- [117] G. van Rossum, “Glue It All Together With Python,” 8 January 1998. [Online]. Available: <https://www.python.org/doc/essays/omg-darpa-mcc-position/>. [Accessed 12 February 2017].
- [118] E. Tollerud and G. Tremblay, “Astropy,” NumFOCUS, 9 January 2017. [Online]. Available: <http://www.astropy.org/>. [Accessed 6 February 2017].
- [119] Sloan Digital Sky Survey Team, “SkyServer,” Alfred P. Sloan Foundation, [Online]. Available: <http://skyserver.sdss.org/dr7/en/tools/search/sql.asp>. [Accessed 29 September 2014].
- [120] R. Lawrence, “The space efficiency of XML,” *Information and Software Technology*, vol. 46, no. 11, pp. 753-759, 2004.
- [121] Sloan Digital Sky Survey Team, “Calibrated Objects,” Alfred P. Sloan Foundation, [Online]. Available: <http://www.sdss.org/dr7/dm/flatFiles/tsObj.html>. [Accessed 12 May 2012].
- [122] Sloan Digital Sky Survey Team, “SDSS DAS Coordinate List Submission Form,” Alfred P. Sloan Foundation, [Online]. Available: http://das.sdss.org/www/html/post_coords.html. [Accessed 24 September 2014].
- [123] Free Software Foundation, Inc., “GNU Wget 1.18 Manual,” 2015. [Online]. Available: <https://www.gnu.org/software/wget/>. [Accessed 19 June 2016].
- [124] Y. Sun, Z. Demirezen, M. Mernik, J. Gray and B. Bryant, ““Is My DSL a Modeling or Programming Language?”,” *Domain-Specific Program Development*, 2008.
- [125] J. V. Ashby, “A comparison of C, Fortran and Java for Numerical Computation,” [Online]. Available: <http://www.stfc.ac.uk/cse/resources/pdf/jaspa.pdf>. [Accessed 26 June 2014].

- [126] W. D. Pence, "CFITSIO User's Guide.", 2006.
- [127] A. P. Smale, R. Brissenden, P. Newman and M. Gibb, "CCFits Documentation," 6 December 2011. [Online]. Available: <http://heasarc.gsfc.nasa.gov/fitsio/CCfits/>. [Accessed 24 September 2014].
- [128] M. Reillo, "JFITSIO Documentation," 11 May 2007. [Online]. Available: jfitsio.sourceforge.net. [Accessed 25 April 2014].
- [129] HEASARC, FITSIO User's Guide, A Subroutine Interface to FITS Format Files, 3.0, Ed., NASA, 2009.
- [130] Space Telescope Science Institute, "PyFits," NASA, 17 July 2014. [Online]. Available: http://www.stsci.edu/institute/software_hardware/pyfits. [Accessed 6 February 2017].
- [131] J.-J. Merelo-Guervós, I. Blancas-Álvarez and P. A. Castillo, "A comparison of implementations of basic evolutionary algorithm operations in different languages," in *Evolutionary Computation (CEC), 2016 IEEE Congress on*, 2016.
- [132] S. W. Haney, "Is c++ fast enough for scientific computing?," *Computers in Physics*, vol. 8, no. 6, pp. 690-696, 1994.
- [133] L. Prechelt, "An empirical comparison of C, C++, Java, Perl, Python, Rexx and Tcl," *IEEE Computer*, vol. 33, no. 10, pp. 23-29, 2000.
- [134] J. Walsh, *private communication*, Dublin: Grid Ireland/TCD, 2008-2013.
- [135] C. Newham, *Learning the bash shell: Unix shell programming*, O'Reilly Media, Inc., 2005.
- [136] A. W. Harris and B. Warner, *A Practical Guide to Lightcurve Photometry and Analysis*, Colorado Springs: Springer, 2006.
- [137] D. P. Schneider, P. B. Hall, G. T. Richards, M. A. Strauss, Vanden Berk, Daniel E., S. F. Anderson, W. N. Brandt, X. Fan, S. Jester, Gray, Jim, Gunn, James E., SubbaRao, Mark U., A. R. Thakar, C. U. Stoughton, Szalay, Alexander S., Yanny, Brian, York,

Donald G, N. Inada, G. R. Knapp, C. M. Krawczyk, Voges, Wolfgang, Bahcall, Neta A., J. Barentine, Blanton, Michael R., Brewington, Howard, Brinkmann, J., R. J. Brunner, F. J. Castander, I. Csabai, Frieman, Joshua A., Fukugita, Masataka, M. Harvanek, D. W. Hogg, Ivezić, Željko, Kent, Stephen M, S. J. Kleinman, G. R. Knapp, Kron, Richard G., J. Krzesinski, D. C. Long, Lupton, Robert H., A. Nitta, Pier, Jeffrey R., Saxe, David H., Y. Shen, Snedden, Stephanie A., Weinberg, David H. and J. Wu, “The Sloan Digital Sky Survey Quasar Catalog. IV. Fifth Data Release,” *The Astronomical Journal*, vol. 134, pp. 102-117, 2007.

- [138] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury and S. Tuecke, “The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets,” *Journal of network and computer applications*, vol. 23, no. 3, pp. 187-200, 2000.
- [139] M. Redmond, “A Review of Coordinates,” [Online]. Available: <http://spiff.rit.edu/classes/phys440/lectures/coords/coords.html>. [Accessed 3 October 2014].
- [140] J. Gray, A. Szalay, M. Nieto-Santisteban and T. Budavari, “SDSS DR10 Navigate Tool,” February 2004. [Online]. Available: <http://skyserver.sdss3.org/dr10/en/tools/chart/navi.aspx>. [Accessed 3 September 2014].
- [141] School of Electrical Engineering and Computer Science, “Top Down Design,” Washington State University, 6 February 2001. [Online]. Available: <http://www.eecs.wsu.edu/~cs150/tdd.htm>. [Accessed 16 November 2014].
- [142] V. Volin, “Design Strategies: Top-Down vs. Bottom-Up,” 13 December 2010. [Online]. Available: <http://superprofundo.com/2010/12/13/top-down-and-bottom-up-pros-and-cons/>. [Accessed 16 November 2014].
- [143] T. Joachims, “Object-Oriented Programming - Lecture 7: Software Design,” 13 February 2012. [Online]. Available: http://www.cs.cornell.edu/courses/cs2110/2012sp/lectures/07-SoftwareDesign_6up.pdf. [Accessed 16 November 2014].

- [144] P. Jalote, *An Integrated Approach to Software Engineering*, New York: Springer, 2005.
- [145] N. Johansson and A. Löfgren, “Designing for Extensibility: An action research study of maximizing extensibility by means of design principles,” University of Gothenburg, Gothenburg, 2009.
- [146] A. Kelly, “The Philosophy of Extensible Software,” ACCU, August 2002. [Online]. Available: <http://accu.org/index.php/journals/391>. [Accessed 17 November 2014].
- [147] D. Hamlet and J. Maybee, *The Engineering of Software: Technical Foundations for the Individual*, Boston: Addison Wesley Longman, 2001.
- [148] R. S. Pressman, *Software Engineering: A Practitioner's Guide*, International ed., New York: McGraw-Hill, 2010.
- [149] F. Detienne, *Software Design – Cognitive Aspect*, F. Bolt, Ed., Springer, 2002.
- [150] C. Larman and V. R. Basili, “Iterative and incremental development: A brief history,” *Computer*, vol. 36, no. 6, pp. 47-56, 2003.
- [151] P. Ralph and Y. Wand, “A proposal for a formal definition of the design concept,” in *Design requirements engineering: A ten-year perspective*, Berlin Heidelberg, Springer, 2009, pp. 103-136.
- [152] B. Thomas, T. Jenness, F. Economou, P. Greenfield, P. Hirst, D. S. Berry, E. Bray, N. Gray, D. Muna, J. Turner, M. de Val-Borro, J. Santander-Vela, D. Shupe, J. Good, G. B. Berriman, S. Kitaeff, J. Fay, O. Laurino, A. Alexov, W. Landry, J. Masters, A. Brazier, R. Schaff, K. Edwards, R. O. Redman, T. R. Marsh, O. Streicher, P. Norris, S. Pascual, M. Davie, M. Droettboom, T. Robitaille, R. Campana, A. Hagen, P. Hartogh, D. Klaes, W. M. Craig and D. Homeier, “Learning from FITS: Limitations in use in modern astronomical research,” *Astronomy and Computing*, vol. 12, pp. 133-145, 2015.
- [153] B. Thomas, T. Jenness, F. Economou, P. Greenfield, P. Hirst, D. Berry, E. Bray, N. Gray, D. Turner, M. de Val-Borro and J. Vela, “Significant problems in FITS limit its use in modern astronomical research,” in *Astronomical Data Analysis Software and*

Systems XXIII, ASP Conference Series, Vol. 485, 2015.

- [154] D. C. Price, B. R. Barsdell and L. J. Greenhill, “Is HDF5 a good format to replace UVFITS?,” in *Astronomical Data Analysis Software and Systems XXIV (ADASS XXIV)*, Calgary, 2015.
- [155] D. C. Price, B. R. Barsdell and L. J. Greenhill, “HDFITS: Porting the FITS data model to HDF5,” *Astronomy and Computing*, vol. 12, pp. 212-220, 2015.
- [156] SDSS, “SDSS Data Archive Server,” [Online]. Available: <http://das.sdss.org/www/html/>. [Accessed 19 February 2016].
- [157] Sloan Digital Sky Survey Team, “Understanding the image processing flags,” Alfred P. Sloan Foundation, 6 December 2006. [Online]. Available: <http://classic.sdss.org/dr7/products/catalogs/flags.html>. [Accessed 24 September 2014].
- [158] Sloan Digital Sky Survey Team, “SDSS Glossary,” Alfred P. Sloan Foundation, 27 June 2007. [Online]. Available: <http://classic.sdss.org/dr7/glossary/index.html>. [Accessed 24 September 2014].
- [159] E. Hickey, *SQL queries to CAS, Private Communication*, 2008.
- [160] H. Wickham, “Tidy Data,” *Journal of Statistical Software*, vol. 59, no. 10, 2014.
- [161] Sloan Digital Sky Survey, “Fourth edition of the SDSS Quasar Catalog,” Alfred P. Sloan Foundation, 9 January 2008. [Online]. Available: http://classic.sdss.org/dr6/products/value_added/qsocat_dr5.html. [Accessed 25 March 2016].
- [162] T. Yonnen and C. Lonvick, “The secure shell (SSH) protocol architecture,” Network Working Group of the IETF, 2006.
- [163] Y. Shafranovich, “Common format and MIME type for Comma-Separated Values (CSV) files,” IETF, 2005.
- [164] W. Pence, “CFITSIO, v2.0: A New Full-Featured Data Interface,” in *Astronomical Data*

- [165] IEEE and The Open Group, “The Open Group Base Specifications,” 2013. [Online]. Available:
http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap03.html.
[Accessed 11 September 2015].
- [166] B. Lin, B. J. Wuebben, C. Wurll and E. Waterlander, “dos2unix - DOS/Mac to Unix and vice versa text file format converter,” 9 June 2014. [Online]. Available:
<http://www.unix.com/man-page/centos/1/unix2dos/>. [Accessed 20 June 2016].
- [167] T. O Briain, *private communication*, Dublin: ITT Dublin, 2007.
- [168] M. Rudenko, T. Spahr, G. Williams, S. Keys, J. L. Galache and J. Davies, “List Of Observatory Codes,” The Minor Planet Centre, 2013. [Online]. Available:
<http://www.minorplanetcenter.net/iau/lists/ObsCodesF.html>. [Accessed 2 September 2014].
- [169] D. Grennan, *Data Specification of Observations from Raheny Observatory*, Dublin, 2014.
- [170] X. Fan, “Simulation of stellar objects in sdss color space,” *The Astronomical Journal*, vol. V, no. 117, p. 2528, 1999.
- [171] C. W. Allen, *Astrophysical Quantities*, 3rd ed., London: Athlone: Springer, 1976.
- [172] Sloan Digital Sky Survey, “The Scope of DR10,” SDSS-III, 2013. [Online]. Available:
<https://www.sdss3.org/dr10/scope.php>. [Accessed 13 June 2016].
- [173] D. G. York, J. Adelman, J. E. J. Anderson, S. F. Anderson, J. Annis, N. A. Bahcall, J. A. Bakken, R. Barkhouser, S. Bastian, E. Berman, W. N. Boroski, S. Bracker, C. Briegel, J. W. Briggs, Brinkmann, J., Brunner, Robert, Burles, Scott, Carey, Larry, Carr, Michael A., Castander, Francisco J., Chen, Bing, Colestock, Patrick L., Connolly, A. J., Crocker, J. H., Csabai, István, Czarapata, Paul C., Davis, John Eric, Doi, Mamoru, Dombeck, Tom, Eisenstein, Daniel, Ellman, Nancy, Elms, Brian R., Evans, Michael L., Fan,

Xiaohui, Federwitz, Glenn R., Fiscelli, Larry, Friedman, Scott, Frieman, Joshua A., Fukugita, Masataka, Gillespie, Bruce, Gunn, James E., Gurbani, Vijay K., de Haas, Ernst, Haldeman, Merle, Harris, Frederick H., Hayes, J., Heckman, Timothy M., Hennessy, G. S., Hindsley, Robert B., Holm, Scott, Holmgren, Donald J., Huang, Chihao, Hull, Charles, Husby, Don, Ichikawa, Shin-Ichi, Ichikawa, Takashi, Ivezić, Željko, Kent, Stephen, Kim, Rita S. J., Kinney, E., Klaene, Mark, Kleinman, A. N., Kleinman, S., Knapp, G. R., Korienek, John, Kron, Richard G., Kunszt, Peter Z., Lamb, D. Q., Lee, B., Leger, R. French, Limmongkol, Siriluk, Lindenmeyer, Carl, Long, Daniel C., Loomis, Craig, Loveday, Jon, Lucinio, Rich, Lupton, Robert H., MacKinnon, Bryan, Mannery, Edward J., Mantsch, P. M., Margon, Bruce, McGehee, Peregrine, McKay, Timothy A., Meiksin, Avery, Merelli, Aronne, Monet, David G., Munn, Jeffrey A., Narayanan, Vijay K., Nash, Thomas, Neilsen, Eric, Neswold, Rich, Newberg, Heidi Jo, Nichol, R. C., Nicinski, Tom, Nonino, Mario, Okada, Norio, Okamura, Sadanori, Ostriker, Jeremiah P., Owen, Russell, Pauls, A. George, Peoples, John, Peterson, R. L., Petravick, Donald, Pier, Jeffrey R., Pope, Adrian, Pordes, Ruth, Prosapio, Angela, Rechenmacher, Ron, Quinn, Thomas R., Richards, Gordon T., Richmond, Michael W., Rivetta, Claudio H., Rockosi, Constance M., Ruthmansdorfer, Kurt, Sandford, Dale, Schlegel, David J., Schneider, Donald P., Sekiguchi, Maki, Sergey, Gary, Shimasaku, Kazuhiro, Siegmund, Walter A., Smee, Stephen, Smith, J. Allyn, Snedden, S., Stone, R., Stoughton, Chris, Strauss, Michael A., Stubbs, Christopher, SubbaRao, Mark, Szalay, Alexander S., Szapudi, Istvan, Szokoly, Gyula P., Thakar, Anirudda R., Tremonti, Christy, Tucker, Douglas L., Uomoto, Alan, Vanden Berk, Dan, Vogeley, Michael S., Waddell, Patrick, Wang, Shu-i., Watanabe, Masaru, Weinberg, David H., Yanny, Brian and Yasuda, Naoki, “The sloan digital sky survey: Technical summary,” *The Astronomical Journal*, vol. 120, no. 3, p. 1579, 2000.

[174] J. Ryan, *private communication*, Dublin, 2007-2009.

[175] Sloan Digital Sky Survey, “Resolve,” Alfred P. Sloan Foundation, 2016. [Online]. Available: <http://www.sdss.org/dr12/algorithms/resolve/>. [Accessed 16 June 2016].

[176] G. Fox and D. Gannon, “Using Clouds for Technical Computing,” in *Cloud Computing and Big Data*, C. Catlett, W. Gentzsch, L. Grandinetti, G. Joubert and J. L. Vasques-

Poletti, Eds., Amsterdam, IOS Press BV, 2013, pp. 81-102.

- [177] K. Hwang, J. Dongarra and G. C. Fox, Distributed and cloud computing: from parallel processing to the internet of things, Waltham: Morgan Kaufmann, 2013.
- [178] D. O'Callaghan, *private communication*, Dublin: Grid Ireland/TCD, 2011-2013.
- [179] R. Lambert, "Grid and data management," TWiki, 4 June 2013. [Online]. Available: <https://twiki.cern.ch/twiki/bin/view/LHCb/GridAndDataManagement>. [Accessed 2 October 2014].
- [180] Amazon Web Services, Inc., "Amazon Simple Storage Service (S3) - Online Cloud Storage for Data and Files," Amazon Web Services, Inc., 2014. [Online]. Available: <http://aws.amazon.com/s3/>. [Accessed 2 October 2014].
- [181] D. G. Monet, S. E. Levine, B. Canzian, H. D. Ables, A. R. Bird, C. C. Dahn, H. H. Guetter, H. C. Harris, A. A. Henden, S. K. Leggett, H. F. Levison, C. B. Luginbuhl, J. Martini, A. K. B. Monet, Munn, Jeffrey A., Pier, Jeffrey R., Rhodes, Albert R., Riepe, Betty, Sell, Stephen, Stone, Ronald C., Vrba, Frederick J., Walker, Richard L., Westerhout, Gart, Brucato, Robert J., Reid, I. Neill, Schoening, William, Hartley, M., Read, M. A. and Tritton, S. B., "The USNO-B Catalog," *The Astronomical Journal*, vol. 125, no. 2, pp. 984-993, 2003.
- [182] J. Murty, Programming amazon web services: S3, EC2, SQS, FPS, and SimpleDB, O'Reilly Media, Inc., 2008.
- [183] Amazon Web Services Inc., "Amazon Web Services (AWS) - Cloud Computing Services," 2014. [Online]. Available: <http://aws.amazon.com/>. [Accessed 24 September 2014].
- [184] Amazon Web Services Inc., "Large Scale Computing and Huge Data Sets," [Online]. Available: http://media.amazonwebservices.com/architecturecenter/AWS_ac_ra_largescale_05.pdf. [Accessed 24 September 2014].

- [185] Amazon Web Services Inc., “Sloan Digital Sky Survey DR6 Subset: Public Data Sets,” 20 January 2012. [Online]. Available: <https://aws.amazon.com/datasets/2797>. [Accessed 9 September 2014].
- [186] P. Doyle, *private communication*, Dublin: DIT, 2012-2014.
- [187] V. Hindriksen, “Stream Computing,” 16 July 2012. [Online]. Available: <http://streamcomputing.eu/blog/2012-07-16/how-expensive-is-an-operation-on-a-cpu/>. [Accessed 26 July 2014].
- [188] Peterson, “The Math Forum,” Drexel University, 16 January 2002. [Online]. Available: <http://mathforum.org/library/drmath/view/51836.html>. [Accessed 1 October 2014].
- [189] M. Hareter, P. Reegen, R. Kuschnig, W. W. Weiss, J. M. Matthews, S. M. Rucinski, D. B. Guenther, A. F. J. Moffat, D. Sasselov and G. A. H. Walker, “Data Reduction pipeline for MOST Guide Stars and Application to two Observing Runs,” *Communications in Asteroseismology*, vol. 156, pp. 48-72, 2008.
- [190] L. A. Buchhave, M. Bizzarro, D. W. Latham, D. Sasselov, W. D. Cochran, M. Endl, H. Issacson, D. Juncher and G. W. Marcy, “Three regimes of extrasolar planets inferred from host star metallicities,” *Nature*, vol. 509, no. 7502, pp. 593-595, 2014.
- [191] D. Segransan, P. Kervella, T. Forveille and D. Queloz, “First radius measurements of very low mass stars with the VLTI,” *Astronomy and Astrophysics*, vol. 397, no. 3, pp. 5-8, 2003.
- [192] W. H. S. Monck, “The Photo-Electric Effect of Starlight,” *Astronomy and Astrophysics*, vol. 11, p. 843, 1892.
- [193] R. Gal and S. Jester, “SDSS Sky Coverage notes for DR5,” Xeno Media, 11 June 2008. [Online]. Available: <http://www.sdss2.org/dr6/>. [Accessed 6 August 2014].
- [194] National Solar Observatory – Sacramento Peak, “Magnitude,” 4 November 1997. [Online]. Available: <http://www.nso.edu/PR/answerbook/magnitude.html>. [Accessed 6 February 2008].

- [195] J. Bryant, S. E. and P. S. Bunclark, “Astronomy, Data and the Problem of Large File Transfers,” in *Astronomical Data Analysis Software and Systems XVII. Vol. 394.*, 2008.
- [196] A. Odlyzko, “The many paradoxes of broadband,” *First Monday*, vol. 8, no. 9, 2003.
- [197] Sloan Digital Sky Survey Team, “The Tenth SDSS Data Release (DR10) - SDSS-III,” SDSS-III, 2010. [Online]. Available: <https://www.sdss3.org/dr10/>. [Accessed 24 September 2014].
- [198] A. Huang, “education, Comparison of programming performance: Promoting STEM and computer science,” in *Integrated STEM Education Conference (ISEC)*, 2015.

A-b SDSS Acknowledgement

This project made use of the SDSS Catalogue

Funding for the SDSS and SDSS-II has been provided by the Alfred P. Sloan Foundation, the Participating Institutions, the National Science Foundation, the U.S. Department of Energy, the National Aeronautics and Space Administration, the Japanese Monbukagakusho, the Max Planck Society, and the Higher Education Funding Council for England. The SDSS Web Site is <http://www.sdss.org/>.

The SDSS is managed by the Astrophysical Research Consortium for the Participating Institutions. The Participating Institutions are the American Museum of Natural History, Astrophysical Institute Potsdam, University of Basel, University of Cambridge, Case Western Reserve University, University of Chicago, Drexel University, Fermilab, the Institute for Advanced Study, the Japan Participation Group, Johns Hopkins University, the Joint Institute for Nuclear Astrophysics, the Kavli Institute for Particle Astrophysics and Cosmology, the Korean Scientist Group, the Chinese Academy of Sciences (LAMOST), Los Alamos National Laboratory, the Max-Planck-Institute for Astronomy (MPIA), the Max-Planck-Institute for Astrophysics (MPA), New Mexico State University, Ohio State University, University of Pittsburgh, University of Portsmouth, Princeton University, the United States Naval Observatory, and the University of Washington.

In addition to the works cited in particular cases above, the following works upon which SDSS is built were used at various points during the project

Blanton, M.R., Lin, H., Lupton, R.H., Maley, F.M., Young, N., Zehavi, I., and Loveday, J. 2003, AJ, 125, 2276

Fukugita, M., Ichikawa, T., Gunn, J.E., Doi, M., Shimasaku, K., and Schneider, D.P. 1996, AJ, 111, 1748

Gunn, J.E., Carr, M.A., Rockosi, C.M., Sekiguchi, M., et al. 1998, AJ, 116, 3040

Gunn, J.E., Siegmund, W.A., Mannery, E.J., Owen, R.E., et al. 2006, AJ, 131, in press (astro-ph/0602326)

- Hogg, D.W., Finkbeiner, D.P., Schlegel, D.J., and Gunn, J.E. 2001, AJ, 122, 2129
- Ivezic, Z., Lupton, R.H., Schlegel, D., et al. 2004, AN, 325, 583
- Lupton, R. 2006, AJ, submitted
- Lupton, R.H., Gunn, J.E., and Szalay, A.S. 1999, AJ, 118, 1406
- Padmanabhan, N., et al. 2008, ApJ, 674, 1217
- Pier, J.R., Munn, J.A., Hindsley, R.B., Hennessy, G.S., Kent, S.M., Lupton, R.H., and Ivezic, Z. 2003, AJ, 125, 1559
- Richards, G.T., Fan, X., Newberg, H., et al. 2002, AJ, 123, 2945
- Smith, J.A., Tucker, D.L., Kent, S.M., et al. 2002, AJ, 123, 2121
- Tucker, D., Kent, S., Richmond, M.W., et al. 2006, AN, 327, 821
- York, D.G., Adelman, J., Anderson, J.E., et al. 2000, AJ, 120, 1579

Appendix B Symbols and Terminology

A number of structural and stylistic conventions are followed within this document. These are outlined here for reference.

B-a Hierarchy of the Thesis

This Thesis is divided into a hierarchical structure. Four key terms are used to define this structure. The highest level is referred to as a *Part*, a conceptual structure which refers to a grouping of several Chapters as laid out in 1.2. Parts are denoted by Roman Numerals e.g. Part I. Beneath the Parts, each numbered *Chapter* addresses a specific aspect of the project, such as Chapter 2 – Core Concepts of Photometric Variability. Each Chapter is divided into Sections, which are identified by the Chapter number followed by the Section number within that Chapter, e.g. Section 1.2. This convention is followed further with Subsections which may be subordinate to a Section (e.g. Subsection 1.2.3) or to another Subsection (e.g. 1.2.3.4.)

This hierarchy dictates a Top-Down, depth-first approach to each topic in the Thesis. In each Chapter, an overview of the topic as a whole is given first. Then, each subject within that topic is probed completely in its own Section, with layers of Subsections as needed until the subject is exhausted before moving on to the next subject, and the next Section.

B-b Text Conventions

A number of text styles and conventions are used throughout this Thesis. Each is used for a specific purpose, as described below.

Citations and references to research documents are given in short parenthetical form inline e.g. (Creaner et al., 2010) and using endnotes identified with square brackets text: e.g. [3]. The parenthetical form is preferred for direct quotes.

In electronic copies of this Thesis, cross-references to Diagrams, Chapters etc are typically hyperlink enabled. These may be activated by the usual method for following hyperlinks for the system the reader is using.

Where the word "and" forms part of a heading, the word is replaced by the ampersand symbol "&." The Ampersand is not used in other parts of the Thesis except in code excerpts where it is used as part of a programming language.

Font	Purpose
Times New Roman, 12 point	Body Text
Times New Roman, 16 point, Bold	Chapter Titles
Times New Roman, 12 point, Bold.	Section and Lower Headings
Times New Roman, 10 point, Bold	Captions for Tables, Figures etc.
Arial, 10 point	Table or figure body text
white text	Dark Backgrounds
Courier new, 12 point	Code excerpts
Half sized text	Documents recreated in figures
<i>italics</i>	Emphasis

Figure 15-1: Table of Text Styles

B-c Design Symbols

Below are definitions of the symbols used in all design documents throughout the project, as well as a description of their use. These symbols and terms are used throughout this Thesis

Throughout the project and this Thesis, a given combination of shape and colour refer to objects of the same type, for example orange rectangles refer to programs and scripts. Within that scope, different shades of the same colour refer to different places within a hierarchy. For example, the Data Access API, as defined in Section 7.2.1, is a script which calls two programs, Diagnose (7.2.1.1) and Extract (7.2.1.2), within it. The lighter shade would be used to denote the API as the calling program, while the two called programs would be described in darker colours. Should a further call be made within those programs, darker and darker shades would be used.

B-c-a Programs and Scripts

Orange rectangles indicate programs and scripts in the design documents. Programs and scripts are the smallest component of the project that may be run on their own. Each program or script may have arguments which are needed for the program to run

and may have data inputs and outputs. Programs typically include one or more functions and one or more processes, and may call other programs.



Figure 15-2: Programs & Scripts

B-c-b Functions

Green rectangles indicate function calls within the design documents. Functions are significant components of code that are incomplete on their own. A function can be a modular component that can be used at several stages of the project, or may be a highly specialised function which is only used once. Functions include one or more processes and may call other functions or programs. As it is used in this project, this symbol is not used to indicate calls to functions defined in external libraries such as CFITIO.



Figure 15-3: Functions

B-c-c Processes

Red rectangles indicate individual processes. Processes are individual tasks for the computer, for example a mathematical expression or a data input or output call. Processes comprise one or more lines of code which may call externally defined functions (e.g. CFITSIO functions.)



Figure 15-4: Processes

B-c-d Loops

Yellow rectangles with rounded corners are used to indicate loops or repeating data structures. The nature of the repetition is indicated in the top left corner: The loop

termination condition, or the number of repeated data elements is shown. Nested repetition is indicated by darker shades.

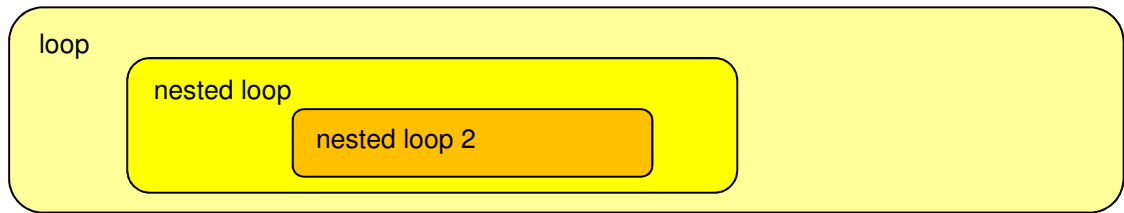


Figure 15-5: Loops & Repeating Data Structures

B-c-e Logical Operations

Magenta diamonds are used to indicate single logical operations. Such logical operations will be written within the symbol in short form. This short form is explained in detail in the accompanying text. If both “true” and “false” produce non-trivial results, “true” and “false” cases will be distinguished clearly. In most cases, only the continuation if the result of the logical operation is “true” will be shown.



Figure 15-6: Logical Operations

B-c-f Computing Elements

Computing Elements, either individual machines or collections of machines, are indicated using a horizontal stylised cylinder symbol in black. These elements include Virtual Machines (VMs) and physical computers.



Figure 15-7: Computing Elements

B-c-g Storage Devices

Data storage devices are designated by a vertical stylised cylinder in dark grey. This symbol represents non-volatile data storage systems, such as Hard Disks or other long-

term data storage solutions. It is not used to represent temporary data repositories such as RAM.



Figure 15-8: Storage Devices

B-c-h Stored Data

Data is designated in various shades of blue. Data stored in memory is represented with a parallelogram, while data stored on disk, either in the form of files or directories is represented by the international standard stored data symbol. Directories are shown with a darker shade than files. As can be seen in Chapter 8 there are many layers of directories shown in this project. As a result, the convention of using darker shades to represent deeper components is not used.



Figure 15-9: Stored Data

B-c-i Logical Catalogues

This standard symbol is used to designate entire catalogues of data, by implication in multiple file. The same shade of blue is used as for files. Catalogues have a designated directory structure, which is defined separately. The Catalogue symbol is used to designate the catalogue as a whole.



Figure 15-10: Catalogues

B-c-j Data Flow

The flow of data within a program or function is designated with arrow symbols. These symbols represent variables passed from function to function, or data being written to files or memory locations.

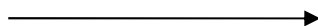


Figure 15-11: Data Flow

B-c-k Logical Operations

Input by the user is designated with an irregular cyan quadrilateral as shown. As part of the project design, this project takes command-line input from the user to vary the parameters and criteria of various programs.

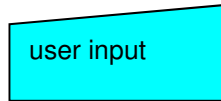


Figure 15-12: User Input

B-c-l Developmental Tools

At various points during the development of the project, work-in-progress versions of programs were designed and developed. At times, it was necessary to designate prospective or planned components of the program with dashed arrows and symbols as shown in Figure 15-13. In addition, during the development process, several versions of a component were tested at the same time. When such components were in testing, they would be designated by a symbol highlighted with a red outline as shown with the function in the diagram.

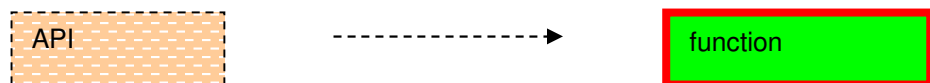


Figure 15-13: Developmental Tools

B-c-m Documents

Documents, such as design documents or even this Thesis itself are denoted by the standard document symbol in white. Individual components of these documents are denoted by increasingly dark shades of grey.

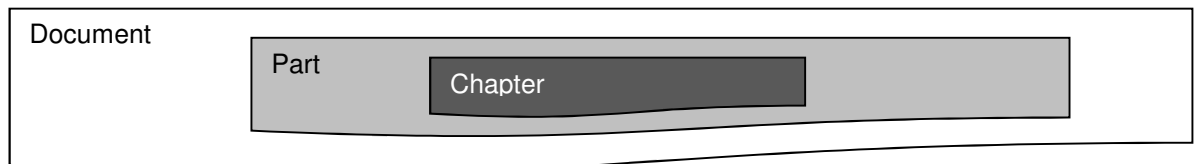


Figure 15-14: Documents

Appendix C Code Sample

In the Subsections below are presented a number of code excerpts from particular scripts, programs or functions as used in the course of this project. These code excerpts are presented as representative examples of each of the programming languages used, and are not exhaustive. Further code samples are available upon request.

C-a C

Shown here is the function `sift` which is used to determine whether a target can be resolved based on the resolution parameter provided by the user as discussed in Subsection 7.2.2.1.1.

```
int sift (struct star **p_input_array,
          long int *p_num_rows,
          long int current_index,
          double resolution,
          double resolution_ra,
          int colour_number)
{
    int sifting_index = 0;
    double mag_difference = 0;
    double dec_difference = 0;

    //for stars higher than the current index, within the
array and
    //within resolution of the current star in RA terms
    for (sifting_index = current_index +1;
        sifting_index < *p_num_rows &&
        resolution_ra > (((*p_input_array) + sifting_index) ->
ra )- (((*p_input_array) + current_index)->ra);
        sifting_index ++)
    {
```

```

        dec_difference =
absolute_difference((( *p_input_array) + sifting_index) ->
dec ), (( *p_input_array) + current_index)->dec));

        //if they are also within 5 FWHM of the current
star in
        //dec terms and within 5 magnitudes of it.
        if (resolution > dec_difference )
        {
            mag_difference = absolute_difference
(( *p_input_array) + sifting_index) -> mag[colour_number] ,
(( *p_input_array) + current_index) -> mag[colour_number]);
            if(mag_difference < 5)
            {
                //sets both stars to error values to be
filtered
                //at the next step
                (( *p_input_array) + sifting_index) -
>rating = 0;
                (( *p_input_array) + current_index) -
>rating = 0;
            }
        }
    }
    return (ERR_NO_ERROR);
}

```

C-b BASH

Shown here is the Bash script `call_xop_6_5.sh` which is used to run the Exoplanet job on the grid as discussed in Subsections 7.2.2.1 and 7.2.4.

```
#!/bin/bash -vx
```

```
#this is a script to call the Exoplanet stage of the  
project
```

```
#call_xop_6_5.sh
```

```
#Version 6.5
```

```
#13th December 2011
```

```
#this script carries out the diagnose and extract  
executables on the parameter file
```

```
#specified in the first argument, $1. and the text file  
specified as argument $2
```

```
#it uses the colour specified with a character in the third  
argument
```

```
#and copies the results to a directory in the LFC speficied  
by the fourth
```

```
#It copies the executables, the parameter file
```

```
#the fits files and other relevant data to the node and  
carries out the executables
```

```
#it then copies and registers theoutput fits files to the  
LFC and cleans up
```

```
#names the parameter file fromt the first argument for  
clarity of reading
```

```
PPR_FILE=$1
```

```
PRT_FILE=$2
```

```
COLOUR=$3
```

```
RUN_NAME=$4
```

```
DEBUG="no"
```

```
if [ "x${DEBUG}" = "xyes" ]; then
```

```
    echo $PPR_FILE
```

```
    echo $PRT_FILE
```

```

fi

#defines cosmo root directory
COSMO_ROOT=/grid/cosmo

#defines the root of the project directory structure
PROJ_ROOT=ittd/grid_cdm

#creates directories if no directories exist
if [ ! -d ${PROJ_ROOT} ]; then
    mkdir -p ${PROJ_ROOT}
fi

LOCUS_ALGORITHM_DIR="${PROJ_ROOT}/release/locus_algorithm"
if [ ! -d ${LOCUS_ALGORITHM_DIR} ]; then
    mkdir -p ${LOCUS_ALGORITHM_DIR}
fi

PPR_DIR=`dirname ${PPR_FILE}`
if [ ! -d ${PPR_DIR} ]; then
    mkdir -p ${PPR_DIR}
fi

# copies diagnose executable from permanent () to temporary
(test)
lcp-cp                                --vo                                cosmo
lfn:${COSMO_ROOT}/${LOCUS_ALGORITHM_DIR}/locus_algorithm_6_5
file:${LOCUS_ALGORITHM_DIR}/locus_algorithm_6_5

#and makes it executable
chmod 755 ${LOCUS_ALGORITHM_DIR}/locus_algorithm_6_5

```

cccX


```
#copies Parameter prm and txt files from permanent  
(workspace/debug) to temporary (test)
```

```
lcp-cp --vo cosmo lfn:$COSMO_ROOT/$PPR_FILE file:$PPR_FILE
```

```
lcp-cp --vo cosmo lfn:$COSMO_ROOT/$PRT_FILE file:$PRT_FILE
```

```
#mkdir?
```

```
#this line gets the number of lines in the temp file
```

```
#this corresponds to the number of files in the directory
```

```
NUM_LINES=`wc $PRT_FILE | awk '{print $1}'`
```

```
NAME=Stranger
```

```
P=p
```

```
CP_FAIL_COUNT=0
```

```
#due to the layout of prm text files, it is necessary to  
use lines 3 onward
```

```
for (( NAME_INDEX = 3 ;    $NAME_INDEX <= $NUM_LINES+1;  
NAME_INDEX++ ))
```

```
do
```

```
#this line gets the filename from the temp file and stores  
it in the variable name.
```

```
#this name is the name of the PPR and TXT files
```

```
PRT_FITS_NAME=`sed -n $NAME_INDEX$P $PRT_FILE`
```

```
#this line gets the directory the original file is in
```

```
PRT_FITS_DIR_NAME=`dirname $PRT_FITS_NAME`
```

ccxi

```

#this line gets the root of the filename
PRT_FITS_NAME_BASE=`basename ${PRT_FITS_NAME} .fit`

#this line makes the name fo the local file
BASE_LOCAL_FITS_NAME=${PRT_FITS_NAME_BASE}_local.fit

#this line makes the local catalogue directory name
LFC_FITS_LOCAL_DIR_NAME=`echo
${COSMO_ROOT}/${PRT_FITS_DIR_NAME} | sed 's/raw/local/g'`

#this line si included even though it is unnecessary so the
code is easier to interpres
NODE_FITS_DIR_NAME=${PRT_FITS_DIR_NAME}

#this line makes the name fo the local file
NODE_FITS_NAME=${NODE_FITS_DIR_NAME}/${BASE_LOCAL_FITS_NAME
}

#this line makes the filename it is to be stored in in the
LFC
LFC_FITS_NAME=${LFC_FITS_LOCAL_DIR_NAME}/${BASE_LOCAL_FITS_
NAME}

#use mkdir and ls to make a dir if it isn't there
ls -d ${NODE_FITS_DIR_NAME} &> /dev/null
test_dir_exists=$?
if [ x"${test_dir_exists}" != x"0" ]; then
    mkdir -p ${NODE_FITS_DIR_NAME}
fi

```

```

#this line copies the named file (NAME)from source to
destination
lcg-cp          --vo          cosmo          lfn:${LFC_FITS_NAME}
file:${NODE_FITS_NAME}
lcg_cp_status=$?
  if [ x"${lcg_cp_status}" != x"0" ]; then
    echo      "lcg-cp      --vo      cosmo      lfn:${LFC_FITS_NAME}
file:${NODE_FITS_NAME} failed with error ${lcg_cp_status}"
    CP_FAIL_COUNT=`expr ${CP_FAIL_COUNT} + 1 `
  fi

done

#executes algorithm
#and redirect its output to null
./${LOCUS_ALGORITHM_DIR}/locus_algorithm_6_5      ${PPR_FILE}
${COLOUR} &>/dev/null

LOCUS_ALGORITHM_EXIT=$?

LFC_OUT_DIR_NAME=${COSMO_ROOT}/${PROJ_ROOT}/data/output/${R
UN_NAME}

#this line gets the directory the file is in
PPR_DIR_NAME=`dirname $PPR_FILE`

#this line gets the root of the filename
PPR_NAME_BASE=`basename ${PPR_FILE} .ppr`

#this makes the short name of the output file

```

```

OUT_NAME=${PPR_NAME_BASE}_out.fit

#this line makes the full name fo the local file
NODE_OUT_NAME=${PPR_DIR_NAME}/${OUT_NAME}

#this line makes the filename it is to be stored in in the
LFC

LFC_OUT_NAME=`echo ${LFC_OUT_DIR_NAME}/${OUT_NAME}`

#use lfc-mkdir and lfc-ls to make a dir if it isn't there
lfc-ls -d ${LFC_OUT_DIR_NAME}
lfc_test_dir_exists=$?
if [ x"${lfc_test_dir_exists}" != x"0" ]; then
    lfc-mkdir -p ${LFC_OUT_DIR_NAME}
fi

#sets up a counter for the number of failed copy and
register results
CR_FAIL_COUNT=0

#use lfc-ls to make see if the file already exists
lfc-ls ${LFC_OUT_NAME}
lfc_test_file_exists=$?
if [ x"${lfc_test_file_exists}" != x"0" ]; then
    lcg-cr          --vo      cosmo      -l      lfn:${LFC_OUT_NAME}
file:${NODE_OUT_NAME}
    lcg_cr_status=$?
    if [ x"${lcg_cr_status}" != x"0" ]; then
        echo "lcg-cr          --vo      cosmo      -l      lfn:${LFC_OUT_NAME}
file:${NODE_OUT_NAME} failed with error ${lcg_cr_status}"
        CR_FAIL_COUNT=`expr ${CR_FAIL_COUNT} + 1`
    fi
fi

```

```

#displays the various possible failure states of the
program at the end
echo "Locus Algorithm exit code ${LOCUS_ALGORITHM_EXIT}"
>logfile.out
echo "Number of failed lgc-cp commands ${CP_FAIL_COUNT}" >>
logfile.out
echo "Number of failed lgc-cr commands ${CR_FAIL_COUNT}" >>
logfile.out

#checks the error states and returns an error if needed.

if [ x"${LOCUS_ALGORITHM_EXIT}" != x"0" ]; then
    exit ${LOCUS_ALGORITHM_EXIT}
fi
if [ x"${CR_FAIL_COUNT}" != x"0" ]; then
    exit ${CR_FAIL_COUNT}
fi
if [ x"${CP_FAIL_COUNT}" != x"0" ]; then
    exit ${CP_FAIL_COUNT}
fi

exit 0

```

C-c JDL

Shown here is a sample JDL file `two_4_950_1000.ppr` which was used to describe one of 1,791 grid jobs submitted as part of the Exoplanet job.

```
Executable = "call_xop_6_5.sh";
```

```

StdOutput = "std.out";
StdError = "std.err";
#Arguments = "two_4_950_1000.ppr two_4_950_1000.txt r 31-1-
2012/0_25-0_000277-2-0_1/r/";
Arguments = "ittd/grid_cdm/data/parameters/ppr/13-1-
2012/0_25-0_000277-2-0_1/two_4_950_1000.ppr
ittd/grid_cdm/data/parameters/ppr/13-1-2012/0_25-0_000277-
2-0_1/two_4_950_1000.txt r 31-1-2012/0_25-0_000277-2-
0_1/r/";
OutputSandbox = {"std.out","std.err","logfile.out"};
InputSandbox =
{"ittd/grid_cdm/scripts/xop/call_xop_6_5.sh"};
#InputSandbox =
{"ittd/grid_cdm/scripts/xop/call_xop_6_5.sh","ittd/grid_cdm
/data/parameters/ppr/13-1-2012/0_25-0_000277-2-
0_1/two_4_950_1000.ppr",
"ittd/grid_cdm/data/parameters/ppr/13-1-2012/0_25-0_000277-
2-0_1/two_4_950_1000.txt"};
VirtualOrganisation = "cosmo";
Requirements = other.GlueCEUniqueID ==
"gridgate.scg.nuigalway.ie:2119/jobmanager-pbs-threeday" ||
other.GlueCEUniqueID ==
"gridgate.cs.tcd.ie:2119/jobmanager-pbs-threeday" ||
other.GlueCEUniqueID == "gridgate.ucc.ie:2119/jobmanager-
pbs-threeday";
OutputSandbox = {"std.out","std.err","logfile.out"};

```

C-d SQL

Shown below is the SQL script `find_reference_stars.sql` which is used to identify the coordinates for the reference stars for a given pointing and target, in this case, the pointing for SDSS J203733.62+001953.5, the target with the highest score in the sample of 10^6 targets from the Exoplanet Catalogue.

```

--sql query to locate reference stars for an entry in the
output catalogue, given pointing and target info
--variables used for SDSS J203733.62+001953.5 (top target
in sample)
--declare pointing coordinates
declare @pointing_ra float;
set @pointing_ra = 309.4971506647;
declare @pointing_dec float;
set @pointing_dec = 0.2737432692606;

--declare target g, r, i
declare @target_g float;
set @target_g = 18.15453338623;
declare @target_r float;
set @target_r = 17.69709205627;
declare @target_i float;
set @target_i = 17.52318382263;

--declare fov size
declare @fov_size float;
set @fov_size = 0.25;
declare @fov_size_ra float;
set @fov_size = 0.25/cos(@pointing_dec); --calculate fov
size in RA

--declare mag match limit
declare @mag_lim float;
set @mag_lim = 2.0;

--declare colour match limit
declare @col_lim float;
set @col_lim = 0.1;

```

```

select ra, dec from photoobjall
  where ( --Local Catalogue clean stars
          mode = 1
          AND type = 6
          AND((flags_r & 0x10000000) != 0) --good flags
          AND ((flags_r & 0x80020) = 0) --bad flags_1
          AND ((CAST((flags_r/0x100000000) as INT) &
0x8100) = 0 )--bad flags_2
        )
    -- in FoV
    AND (ra<(@pointing_ra+(@fov_size_ra/2)))AND
(ra>(@pointing_ra-(@fov_size_ra/2)))
    AND (dec<(@pointing_dec+(@fov_size/2)))AND
(dec>(@pointing_dec-(@fov_size/2)))

    -- mag match
    AND (r<(@target_r+@mag_lim))AND (r>(@target_r-
@mag_lim))

    -- col match
    AND ((g-r)<((@target_g-@target_r)+(@col_lim)AND ((g-
r)>((@target_g-@target_r)-(@col_lim)
    AND ((r-i)<((@target_r-@target_i)+(@col_lim)AND ((r-
i)>((@target_r-@target_i)-(@col_lim)

```


Appendix D Results Samples

A number of short excerpts from the various files and catalogues are provided here for reference

D-a Distribution of SDSS flags

The flags for a sample of 62630 stars in the SDSS catalogue are shown in Table 15-1, Table 15-2, and Table 15-3. These distributions were generated by taking an arbitrary sample of data from SDSS and processing it into individual flags using MS-Excel.

FLAGS_1	Count	Percentage
CANONICAL_CENTER	0	0.00%
BRIGHT	3420	5.46%
EDGE	3440	5.49%
BLENDED	12960	20.69%
CHILD	16843	26.89%
PEAKCENTER	286	0.46%
NODEBLEND	7275	11.62%
NOPROFILE	0	0.00%
NOPETRO	47520	75.87%
MANYPETRO	11657	18.61%
NOPETROBIG	0	0.00%
DEBLEND_TOO_MANY_PEAKS	77	0.12%
CR	8928	14.26%
MANYR50	0	0.00%
MANYR90	0	0.00%
BAD_RADIAL	0	0.00%
INCOMPLETE_PROFILE	0	0.00%
INTERP	26165	41.78%
SATUR	3148	5.03%
NOTCHECKED	1598	2.55%
SUBTRACTED	0	0.00%
NOSTOKES	0	0.00%
BADSKY	0	0.00%
PETROFAINT	0	0.00%
TOO_LARGE	0	0.00%
DEBLENDED_AS_PSF	6246	9.97%
DEBLEND_PRUNED	2564	4.09%
ELLIPFAINT	0	0.00%
BINNED1	61932	98.89%
BINNED2	1515	2.42%
BINNED4	357	0.57%
MOVED	12889	20.58%

Table 15-1: Distribution of flags bit in 62630 SDSS fits entries

FLAGS_2	Count	Percentage
DEBLENDED_AS_MOVING	1189	1.90%
NODEBLEND_MOVING	7555	12.06%
TOO_FEW_DETECTIONS	2201	3.51%
BAD_MOVING_FIT	4470	7.14%
STATIONARY	48457	77.37%
PEAKS_TOO_CLOSE	4037	6.45%
BINNED_CENTER	0	0.00%
LOCAL_EDGE	0	0.00%
BAD_COUNTS_ERROR	0	0.00%
BAD_MOVING_FIT_CHILD	1722	2.75%
DEBLEND_UNASSIGNED_FLUX	519	0.83%
SATUR_CENTER	1194	1.91%
INTERP_CENTER	7623	12.17%
DEBLENDED_AT_EDGE	900	1.44%
DEBLEND_NOPEAK	6785	10.83%
PSF_FLUX_INTERP	9080	14.50%
TOO_FEW_GOOD_DETECTIONS	22857	36.50%
CENTER_OFF_AIMAGE	14	0.02%
DEBLEND_DEGENERATE	49	0.08%
BRIGHTEST_GALAXY_CHILD	0	0.00%
CANONICAL_BAND	0	0.00%
AMOMENT_FAINT	0	0.00%
AMOMENT_SHIFT	0	0.00%
AMOMENT_MAXITER	0	0.00%
MAYBE_CR	2742	4.38%
MAYBE_EGHOST	126	0.20%
NOTCHECKED_CENTER	0	0.00%
27	0	0.00%
MEASURED	4	0.01%
GROWN_MERGED	0	0.00%
HAS_CENTER	0	0.00%
MEASURE_BRIGHT	0	0.00%

Table 15-2: Distribution of `flags_2` bit in 62630 SDSS fits entries

STATUS		
SET	62630	100.00%
GOOD	53525	85.46%
DUPLICATE	7633	12.19%
OK_RUN	49323	78.75%
RESOLVED	49323	78.75%
PSEGMENT	49323	78.75%
FIRST_FIELD	0	0.00%
OK_SCANLINE	46403	74.09%
OK_STRIPE	32186	51.39%
SECONDARY	17130	27.35%
PRIMARY	32161	51.35%
TARGET	3372	5.38%

Table 15-3: Distribution of `status` bit in 62630 SDSS fits entries

D-b Local Catalogue Data

Position		Magnitude				
Ra	Dec	<i>u</i>	<i>g</i>	<i>r</i>	<i>i</i>	<i>z</i>
226.7639	2.5577	18.80	16.94	16.17	15.89	15.75
226.7634	2.5596	25.54	21.85	20.36	19.58	19.22
226.7724	2.6026	22.72	22.73	21.63	21.11	20.40
226.7772	2.5441	18.79	16.56	15.69	15.38	15.22
226.8048	2.6495	19.77	18.72	18.51	18.43	18.39
226.8163	2.5217	23.65	21.14	19.70	18.77	18.33
226.8063	2.5153	25.45	25.10	22.30	24.36	21.61
226.8175	2.7015	17.79	16.23	15.64	15.43	15.33
226.8173	2.6724	19.29	17.54	16.84	16.58	16.46
226.8193	2.5936	18.88	18.05	17.71	17.57	17.52
226.8265	2.5613	19.96	18.91	18.42	18.22	18.16
226.8375	2.4936	19.08	18.07	17.64	17.46	17.41
226.8431	2.5740	20.95	18.29	16.85	15.98	15.53
226.8412	2.5769	18.82	17.74	17.33	17.14	17.06
226.8504	2.5801	23.36	20.67	19.19	17.75	16.97
226.8488	2.5789	20.24	19.21	18.90	18.79	18.75
226.8482	2.5749	21.86	20.23	19.58	19.26	19.25
226.8457	2.5826	23.49	21.65	20.16	19.36	19.00
226.8407	2.5782	26.10	23.03	21.82	20.41	19.99
226.8447	2.5705	23.77	24.31	24.78	22.91	20.81
226.8480	2.6235	19.40	18.37	18.00	17.85	17.77
226.8516	2.6637	17.96	16.69	16.21	16.03	15.94
226.8490	2.4946	19.49	16.98	15.89	15.51	15.31
226.8555	2.5743	19.53	18.40	17.93	17.73	17.65
226.8624	2.7002	19.11	17.17	16.38	16.10	15.95
226.8626	2.6332	18.45	17.41	17.10	16.99	16.97
226.8624	2.6007	18.78	17.55	17.10	16.94	16.88
226.8661	2.5724	19.20	18.24	17.81	17.63	17.55
226.8661	2.5630	21.68	19.08	17.63	16.16	15.38
226.8728	2.5287	17.78	16.58	16.69	16.81	16.91
226.8725	2.5307	19.57	18.22	17.70	17.52	17.43
226.8806	2.5130	18.26	16.06	15.18	14.87	14.72
226.8799	2.5123	19.96	18.83	18.38	18.22	18.24
226.8830	2.5125	25.74	23.66	22.16	20.74	19.80
226.8920	2.5051	17.27	15.70	15.13	14.93	14.86
226.9001	2.5214	20.19	19.05	18.70	18.70	18.43
226.9017	2.5206	24.63	25.11	21.32	21.43	22.83
226.8928	2.5152	22.07	20.96	20.48	20.21	20.23
226.9015	2.5242	24.63	25.11	22.48	21.59	22.83
226.9052	2.5191	24.63	21.88	24.59	21.34	22.66
226.9015	2.5235	24.63	25.11	21.91	21.69	21.62
226.9016	2.5224	24.63	25.11	22.01	21.42	22.83
226.9062	2.5191	24.02	22.55	21.73	22.26	21.32
226.9023	2.5582	19.55	17.39	16.56	16.30	16.16
226.7603	2.5592	21.43	19.71	18.87	18.59	18.45
226.7596	2.5336	23.68	20.71	19.27	17.86	17.13
226.7646	2.7022	21.02	20.06	19.62	19.45	19.26
226.7655	2.6959	19.98	18.32	17.65	17.39	17.25
226.7655	2.5735	20.88	19.46	18.89	18.63	18.52
226.7684	2.6177	21.73	18.84	17.39	16.41	15.88

Table 15-4: An arbitrary sample of 50 entries in the Local Catalogue

D-c Output Catalogues

Samples are shown of the top 40 targets by score in the Quasar Catalogue and the top 50 by score in the sample of 10^6 stars from the Exoplanet Catalogue. Also shown is the reference stars that contribute to the pointing for SDSS J203733.62+001953.5, the target with the highest score in the sample of 10^6 targets from the Exoplanet Catalogue

D-c-a Sample of Quasar Catalogue

Position		Magnitude					Pointing		
RA	Dec	<i>u</i>	<i>g</i>	<i>r</i>	<i>i</i>	<i>z</i>	RA	Dec	Score
313.05	-0.45	19.82	19.26	18.65	18.38	17.93	313.00	-0.46	14.14
236.54	6.13	19.22	19.05	18.61	18.43	18.15	236.46	6.20	13.33
255.25	20.15	20.12	19.63	19.26	19.15	19.18	255.31	20.20	13.27
238.76	3.65	19.67	19.49	19.05	18.84	18.89	238.78	3.74	11.08
252.80	25.14	19.53	19.21	18.81	18.62	18.45	252.72	25.06	11.04
116.24	36.92	19.68	19.54	19.08	18.89	18.82	116.33	36.94	10.72
234.34	2.88	19.89	19.75	19.40	19.24	19.25	234.39	2.86	10.72
252.53	34.54	19.10	18.59	18.27	18.16	17.94	252.62	34.49	10.61
248.07	48.37	19.23	18.92	18.57	18.45	18.40	248.09	48.40	9.99
246.31	40.98	19.56	19.56	19.18	19.06	19.18	246.36	40.99	9.94
253.99	32.30	18.38	18.00	17.61	17.52	17.49	254.01	32.31	9.87
247.80	29.16	19.06	19.08	18.65	18.51	18.53	247.83	29.09	9.51
222.30	3.55	20.84	20.62	20.33	20.20	20.15	222.38	3.50	9.48
262.17	55.06	19.12	19.00	18.59	18.43	18.42	262.24	54.99	9.41
254.80	20.62	20.61	20.02	19.29	19.02	18.89	254.88	20.61	9.37
135.06	2.79	20.46	20.04	19.66	19.50	19.34	135.14	2.81	9.34
249.69	41.09	19.30	19.17	18.82	18.69	18.68	249.61	41.01	9.22
246.50	38.55	20.17	19.85	19.32	19.10	19.02	246.44	38.50	9.18
259.69	56.41	19.39	19.34	19.02	18.87	18.89	259.70	56.49	9.14
237.84	3.00	20.26	19.95	19.60	19.43	19.46	237.80	2.98	9.07
232.62	0.93	19.35	18.93	18.58	18.48	18.41	232.55	0.89	9.07
232.65	4.01	19.68	19.57	19.25	19.12	19.15	232.73	3.97	9.02
240.60	2.71	25.02	18.24	17.85	17.72	17.69	240.59	2.75	9.01
117.67	43.43	19.00	18.83	18.45	18.32	18.27	117.78	43.47	8.95
254.00	32.02	21.83	19.49	19.17	19.01	19.06	254.09	32.09	8.88
254.35	20.77	19.45	19.29	19.00	18.87	18.96	254.27	20.85	8.79
135.02	3.93	18.15	18.06	17.75	17.63	17.69	134.94	4.00	8.75
249.98	37.61	19.43	19.31	19.01	18.87	18.80	250.04	37.67	8.75
247.63	29.33	19.60	19.47	19.21	19.09	19.22	247.55	29.28	8.72
188.92	12.61	25.53	20.71	20.09	19.88	19.79	188.91	12.56	8.67
217.13	4.81	20.20	19.64	19.37	19.24	18.97	217.08	4.77	8.67
264.34	55.06	18.91	18.71	18.30	18.11	17.73	264.40	55.13	8.62
215.43	7.01	19.40	19.02	18.67	18.51	18.43	215.40	7.08	8.59
232.07	32.11	19.40	19.05	18.66	18.52	18.46	232.13	32.09	8.57
258.56	57.95	19.29	18.81	18.37	18.22	18.17	258.48	58.04	8.47
220.62	4.55	19.96	19.68	19.28	19.15	19.19	220.63	4.50	8.43
249.88	37.04	19.06	19.05	18.71	18.57	18.53	249.82	37.10	8.37
212.17	-2.25	21.91	19.84	19.39	19.24	19.09	212.10	-2.17	8.35
215.23	59.45	20.53	20.48	20.17	20.05	20.06	215.13	59.51	8.28
148.15	36.13	19.20	19.15	18.87	18.78	18.82	148.23	36.17	8.28

Table 15-5: The top 40 quasars in the Quasar Catalogue

D-c-b Sample of Exoplanet Catalogue

Position		Magnitude					Pointing		
RA	Dec	<i>u</i>	<i>g</i>	<i>r</i>	<i>i</i>	<i>z</i>	RA	Dec	Score
309.39	0.33	19.38	18.15	17.70	17.52	17.47	309.50	0.27	117.72
309.85	-0.14	19.95	18.52	18.02	17.84	17.81	309.83	-0.10	117.56
309.84	-0.01	19.74	18.38	17.87	17.68	17.60	309.83	-0.09	117.16
309.43	0.23	20.01	18.65	18.16	17.99	17.91	309.50	0.27	116.93
309.72	-0.01	19.56	18.34	17.86	17.69	17.61	309.71	0.08	116.73
309.65	0.73	19.07	17.81	17.34	17.16	17.11	309.58	0.65	116.58
309.91	0.00	19.15	17.80	17.30	17.11	17.00	309.83	-0.10	116.54
309.71	-0.15	19.89	18.60	18.10	17.91	17.86	309.83	-0.10	116.21
309.40	0.27	19.77	18.61	18.15	17.98	17.95	309.50	0.25	116.21
309.50	0.63	19.19	17.89	17.42	17.25	17.17	309.58	0.65	116.14
309.78	-0.15	20.01	18.75	18.26	18.09	17.99	309.74	-0.21	116.05
309.60	0.27	19.94	18.67	18.21	18.03	17.93	309.50	0.25	116.01
309.74	0.32	19.93	18.79	18.32	18.15	18.07	309.86	0.19	115.98
309.34	1.04	19.40	18.07	17.58	17.40	17.34	309.44	0.92	115.74
309.74	0.33	20.17	18.90	18.41	18.24	18.18	309.86	0.21	115.26
309.69	-0.10	19.31	17.93	17.43	17.25	17.22	309.82	-0.09	115.24
309.71	-0.07	19.67	18.31	17.79	17.61	17.55	309.83	-0.09	115.24
309.48	-0.60	20.05	18.67	18.14	17.94	17.88	309.43	-0.70	115.06
309.61	0.30	19.50	18.24	17.80	17.63	17.56	309.50	0.27	114.95
309.67	-0.10	19.07	17.80	17.31	17.14	17.09	309.78	-0.10	114.94
309.65	0.77	19.42	18.20	17.73	17.55	17.48	309.58	0.65	114.76
309.61	0.70	19.30	18.06	17.58	17.40	17.33	309.58	0.65	114.73
309.57	0.38	20.23	18.92	18.43	18.26	18.21	309.50	0.27	114.72
309.83	0.25	19.48	18.28	17.82	17.66	17.61	309.72	0.21	114.69
309.49	-0.60	20.12	18.71	18.17	17.97	17.82	309.43	-0.70	114.68
310.68	0.42	19.65	18.36	17.89	17.71	17.63	310.65	0.30	114.67
309.70	-0.19	19.57	18.24	17.74	17.54	17.47	309.83	-0.09	114.64
309.72	0.24	19.75	18.48	17.99	17.82	17.73	309.72	0.15	114.62
309.39	-0.59	19.63	18.37	17.84	17.64	17.56	309.43	-0.70	114.59
309.46	0.41	20.13	18.76	18.30	18.12	18.02	309.50	0.29	114.49
309.98	0.26	19.93	18.58	18.10	17.91	17.83	309.86	0.19	114.45
309.82	-0.01	19.25	18.00	17.51	17.32	17.24	309.83	-0.09	114.39
309.62	-0.09	19.37	18.07	17.60	17.43	17.37	309.74	-0.21	114.38
309.58	0.36	20.06	18.73	18.28	18.11	18.05	309.50	0.25	114.37
310.15	0.81	19.32	18.09	17.61	17.43	17.38	310.08	0.90	114.30
309.57	0.66	19.32	18.11	17.63	17.45	17.39	309.58	0.65	114.16
309.34	0.28	19.93	18.64	18.16	17.98	17.91	309.40	0.25	114.12
309.98	0.30	20.03	18.69	18.20	18.01	17.97	309.86	0.19	113.91
309.82	0.26	19.53	18.20	17.73	17.56	17.49	309.72	0.21	113.82
309.77	-0.14	19.32	18.19	17.69	17.49	17.40	309.83	-0.09	113.75
309.41	0.22	19.39	18.17	17.72	17.56	17.53	309.50	0.27	113.72
310.69	0.31	19.43	18.10	17.62	17.46	17.41	310.64	0.23	113.58
309.66	-0.10	20.10	18.73	18.25	18.08	18.01	309.74	-0.21	113.54
309.85	0.77	19.65	18.38	17.89	17.71	17.70	309.87	0.82	113.54
309.73	-0.11	19.53	18.32	17.86	17.70	17.61	309.74	-0.19	113.50
309.82	0.28	19.59	18.34	17.88	17.70	17.63	309.72	0.21	113.36
310.52	0.30	19.66	18.43	17.96	17.78	17.71	310.65	0.30	113.35
309.47	0.79	19.64	18.31	17.84	17.67	17.60	309.44	0.91	113.32
309.86	-0.20	19.23	18.00	17.52	17.35	17.31	309.83	-0.09	113.25
310.56	0.70	19.43	18.19	17.72	17.55	17.41	310.44	0.82	113.25

Table 15-6: the top 50 stars by score in the sample of 10^6 targets from the Exoplanet Catalogue

D-c-c Reference Stars for SDSS J203733.62+001953.5

RA	Dec	RA	Dec	RA	Dec	RA	Dec	RA	Dec
309.3921	0.3144	309.3801	0.3421	309.4268	0.2207	309.4044	0.2798	309.3693	0.3190
309.3924	0.2865	309.3811	0.3607	309.5267	0.3194	309.4486	0.3943	309.4208	0.3627
309.4655	0.2916	309.3822	0.3217	309.5782	0.2427	309.4499	0.3903	309.4209	0.2948
309.4670	0.1528	309.4355	0.1843	309.3969	0.3986	309.5014	0.3767	309.4706	0.3572
309.5175	0.3597	309.4364	0.2143	309.3973	0.1945	309.5017	0.1980	309.4706	0.3673
309.5179	0.3724	309.4375	0.2174	309.3975	0.2899	309.5521	0.3413	309.5194	0.2898
309.5180	0.3569	309.4875	0.3552	309.3978	0.3891	309.5529	0.2958	309.5197	0.3241
309.5807	0.3319	309.4876	0.3861	309.3979	0.2988	309.6072	0.3807	309.6158	0.3698
309.5810	0.2215	309.4883	0.2068	309.4528	0.1813	309.6087	0.2081	309.6160	0.3638
309.3826	0.3397	309.4883	0.2043	309.5062	0.1596	309.4064	0.1920	309.6161	0.3839
309.3829	0.1837	309.5875	0.1596	309.5069	0.3598	309.4584	0.1970	309.6162	0.1664
309.4317	0.2707	309.4083	0.3413	309.5575	0.2317	309.5085	0.1487	309.6174	0.2546
309.4324	0.2027	309.4090	0.2176	309.6063	0.2972	309.5090	0.3955	309.3758	0.2589
309.4785	0.3906	309.4095	0.3860	309.3995	0.2591	309.5093	0.3895	309.3764	0.2358
309.4786	0.2713	309.5451	0.2768	309.4002	0.3714	309.5097	0.1637	309.3780	0.3379
309.4787	0.1673	309.5458	0.3810	309.4004	0.2867	309.5618	0.1913	309.4290	0.2759
309.4795	0.3260	309.5464	0.3437	309.4910	0.3218	309.5621	0.2785	309.4299	0.1561
309.4798	0.3300	309.3933	0.1598	309.4910	0.3080	309.6095	0.2062	309.4301	0.3880
309.5309	0.3900	309.4424	0.2759	309.4919	0.2002	309.6096	0.2334	309.4302	0.2254
309.5312	0.3125	309.5387	0.2151	309.5503	0.3064	309.6099	0.3768	309.4310	0.2307
309.5815	0.2203	309.5391	0.3830	309.5980	0.3952	309.6104	0.3866	309.4760	0.1588
309.5824	0.3559	309.5922	0.3294	309.5983	0.2043	309.4121	0.2040	309.4762	0.3880
309.3721	0.2969	309.4512	0.2194	309.4022	0.3125	309.4123	0.2558	309.4762	0.3484
309.3723	0.3027	309.5042	0.3817	309.4033	0.1857	309.4130	0.2082	309.4773	0.1564
309.3729	0.1496	309.5051	0.3839	309.4563	0.3175	309.4130	0.2041	309.4778	0.1558
309.3736	0.2567	309.5053	0.1722	309.4982	0.3035	309.4997	0.2913	309.5703	0.1984
309.3737	0.3493	309.5055	0.3987	309.5469	0.1725	309.5005	0.1671	309.5709	0.3939
309.4269	0.2052	309.5615	0.3254	309.5478	0.2037	309.5007	0.2607	309.6199	0.3590
309.4273	0.2106	309.3866	0.3697	309.5479	0.1829	309.5553	0.2741	309.6203	0.1942
309.4273	0.3964	309.3879	0.3714	309.6011	0.2703	309.5555	0.3375	309.3852	0.3491
309.4275	0.2784	309.4849	0.2064	309.6012	0.1814	309.5557	0.2320	309.3862	0.1607
309.4276	0.3636	309.4851	0.2118	309.6015	0.2674	309.6133	0.3531	309.4808	0.2505
309.4276	0.2413	309.4859	0.2509	309.6019	0.3656	309.6134	0.3228	309.4818	0.1993
309.4282	0.2001	309.5337	0.3545	309.6020	0.3304	309.6138	0.1734	309.5761	0.2336
309.4837	0.3320	309.5341	0.3289	309.6024	0.1603	309.6151	0.3521	309.5767	0.3606
309.4843	0.3056	309.5350	0.2406	309.6024	0.3570	309.4151	0.2415	309.6221	0.2480
309.5328	0.3951	309.5353	0.1634	309.6025	0.3265	309.4622	0.3732	309.6222	0.2002
309.3952	0.2323	309.5949	0.3002	309.6026	0.3556	309.5126	0.2900	309.6225	0.3069
309.3954	0.3128	309.5957	0.3951	309.4098	0.3574	309.5135	0.1524	309.6238	0.3838
309.4406	0.3501	309.5965	0.3202	309.4103	0.1877	309.5671	0.3767	309.3748	0.3971
309.4416	0.1943	309.3885	0.3914	309.4111	0.2266	309.5678	0.2951	309.4228	0.3324
309.4421	0.2776	309.3900	0.2860	309.4630	0.3008	309.6126	0.2994	309.4230	0.2032
309.4890	0.1528	309.3901	0.3315	309.4634	0.2525	309.6128	0.1790	309.4236	0.3191
309.4893	0.1752	309.3902	0.3660	309.4636	0.3415	309.6129	0.1837	309.4237	0.3289
309.4902	0.3698	309.4949	0.1546	309.4636	0.2703	309.4690	0.3068	309.5729	0.1847
309.5402	0.2820	309.4950	0.2586	309.5102	0.2132	309.5162	0.3034	309.5730	0.2886
309.5412	0.3787	309.4969	0.2284	309.5551	0.3067	309.5652	0.2653	309.5733	0.1918
309.5418	0.3431	309.5442	0.2039	309.6046	0.2935	309.5654	0.2348		
309.5888	0.3845	309.5989	0.3229	309.4043	0.2709	309.6186	0.2646		
309.5899	0.1586	309.5994	0.3227	309.4044	0.3429	309.3682	0.2699		

Table 15-7: Coordinate (RA, Dec) list for all 247 references for SDSS J203733.62+001953.5, the target with the highest score in the sample of 10⁶ targets from the Exoplanet Catalogue

Appendix E Index

Absolute Photometry, 25

Abstraction, 48

airmass, 32

API, 109, 111, 112, 119, 129, 131, 151, 161, 162, 165, 166, 303

argument, 106, 113, 114, 117, 119, 130, 135, 246, cccx

Assembly Language, 49

Astrometry, 17, 18, 19

Astronomical Catalogues, 38

Bottom-Up, 108, 142

C, iv, 15, 29, 45, 49, 50, 51, 55, 56, 58, 59, 60, 99, 105, 107, 108, 111, 119, 125, 138, 140, 141, 142, 143, 148, 149, 154, 157, 170, 173, 195, 300, cccviii

C++, 50, 51, 55, 56, 99, 173

camcol, 54, 106, 116, 127, 130, 131, 137, 153, 158, 165, 166, 222, 224, 226

Catalogue, 147, 157, 306

Charge Coupled Device, iv, 29

Cloud Computing, 244, 252, 274

colour, iv, 1, 32, 33, 35, 36, 38, 39, 62, 67, 73, 79, 80, 84, 86, 87, 88, 98, 108, 114, 117, 119, 121, 122, 123, 127, 158, 185, 186, 187, 193, 194, 198, 199, 200, 201, 203, 204, 205, 206, 212, 213, 214, 215, 216, 217, 218, 219, 220, 244, 246, 247, 248, 262, 265, 266, 267, 273, 274, 303, cccviii, cccix, cccx, cccxviii

colour index, 36, 86, 87, 123, 127, 186, 193, 194, 212, 213, 214, 215, 216, 217, 246, 247, 248

Comparison-Check Photometry, iv

Cornerpoint, 80

csv, 130, 149, 165

Data Mining, i, 1, 38, 147

Declination, iv, 1, 75, 112, 159, 183, 198, 203

Differential Photometry, 1, 4, 26, 30, 31, 221

Direct Imaging, 17, 18, 19

Elastic Cloud Compute, iv, 255

exoplanet, 17, 110, 152

Extensibility, 91, 94, 101

Extinction, 32

Extrasolar Planets, i, 88

field, vii, 1, 37, 38, 54, 62, 63, 66, 73, 74, 78, 79, 82, 88, 106, 111, 116, 117, 118, 120, 127, 130, 131, 133, 134, 137, 138, 153, 158, 159, 165, 166, 167, 168, 174, 183, 184, 187, 188, 192, 199, 204, 212, 219, 222, 224, 225, 226, 230, 231, 232, 242, 248, 249, 250, 251, 256, 257, 258, 260, 262, 264, 269, 270, 272, 273

Field of View, 1, 37, 76, 188, 198, 201, 248, 261

filter, 36, 86, 99, 100, 114, 115, 121, 123, 153, 158, 159, 183, 185, 186, 193, 200, 203, 224, 246, 248, ccix

FITS, 99, 103, 157

Flexibility, 91, 101

Fortran, 55

gLite, v, 43, 46, 47, 57, 59, 68, 107, 108, 109, 119, 128, 141, 145, 146, 147, 156, 168, 169, 171, 177, 178, 229, 231, 244, 255

grep, 136

Grid Computing, i, 1, 43

Grid Management System, v, 45, 143, 175

HDU, 157

Imperative Language, 50

Interpreter, iv, 58

Java, 49, 55, 56

Job Description Language, v, 46, 106, 164, 168, 169, 171

Job Submission System, v, 45, 107, 145, 253

Language Level, 48

Layering, 94, 101

LFC, 145, 146, 147, 157

LFN, 145

Local catalogue, 159

Local Catalogue, 69, 70, 91, 105, 109, 111, 114, 115, 118, 119, 120, 131, 134, 141, 153, 159, 160, 162, 175, 177, 178, 180, 183, 192, 194, 196, 197, 201, 204, 217, 218, 222, 223, 224, 225, 226, 228, 229, 230, 231, 232, 234, 235, 236, 237, 239, 240, 241, 242, 243, 251, 252, 253, 258, 266, 267, 269, cccxix, cccxxii

Locus, vii, 1, 5, 73, 74, 75, 76, 77, 80, 82, 89, 103, 105, 116, 119, 120, 123, 124, 131, 140, 159, 178, 181, 182, 183, 187, 188, 195, 196, 200, 205, 212, 215, 229, 242, 245, 248, 249, 250, 256, 257, 260, 264, 265, 267, 273, 274, cccxvi

Locus Algorithm, vii, 1, 5, 73, 74, 75, 76, 77, 82, 89, 103, 105, 116, 119, 120, 131, 140, 159, 178, 181, 182, 183, 187, 195, 196, 200, 205, 212, 215, 229, 242, 245, 248, 249, 250, 256, 257, 260, 264, 265, 267, 273, 274, cccxvi

Logical File Catalogue, v, 45, 144, 145

Magnitude, 12, 36, 78, 112, 121, 152, 159, 160, 196, 198, 200, 201, 203, 205, 206, 208, 209, 217, 245, 246, 267, cccxxii, cccxxiii, cccxxiv

Microlensing, 17, 18, 19

Noise, vi, 32, 33, 34, 39

optimised pointing, vii, 5, 35, 64, 71, 187, 191, 195, 198, 205, 230, 231, 237, 244, 250, 251, 273

Output Catalogue, 69, 116, 141, 205, 253, cccxxiii

parameter, 8, 9, 57, 70, 78, 93, 101, 106, 110, 111, 112, 117, 125, 126, 127, 128, 130, 131, 134, 141, 151, 152, 161, 162, 164, 184, 206, 211, 231, 237, 241, 245, 246, 250, 251, 272, cccviii, cccx

Parameter files, 161

Parameterisation, 96, 111, 125, 126, 127, 128, 129, 131, 132, 133, 134, 136, 139, 141, 230, 253

Photometry, iv, 1, 4, 25, 26, 30, 31, 221

Pointing, 82, 124, 190, 198, 199, 200, 203, cccxxiii, cccxxiv

Point-Spread Function, vi, 250

prm, 130, 162

Pulsar Timing, 17

Quasar, 64, 65, 71, 116, 141, 152, 160, 166, 194, 198, 199, 200, 202, 204, 218, 222, 225, 226, 227, 228, 230, 232, 234, 236, 237, 239, 240, 241, 242, 243, 244, 245, 251, 257, 268, 273, 274, cccxxiii

Radial Velocity, 16, 17, 18, 19

rating, 62, 63, 79, 81, 84, 85, 86, 87, 88, 89, 94, 120, 121, 122, 123, 186, 189, 190, 193,
212, 233, 246, 247, 248, 265, 266, 267, cccix

Reference Stars, 183, cccxxv

Relative Photometry, 26

rerun, 54, 106, 116, 127, 130, 131, 137, 158, 165, 166, 222, 224, 226

Right Ascension, vi, 1, 75, 76, 112, 159, 183, 198, 203

run, 43, 49, 53, 54, 56, 58, 71, 105, 106, 107, 110, 111, 116, 118, 126, 127, 130, 131,
135, 136, 137, 141, 143, 146, 150, 153, 158, 165, 166, 168, 173, 178, 182, 222, 224,
226, 228, 229, 232, 233, 234, 237, 238, 241, 242, 243, 267, 269, 274, 303, cccix

Scintillation, 33

score, vii, 1, 62, 63, 66, 67, 71, 74, 81, 82, 83, 84, 88, 89, 121, 125, 160, 174, 178, 181,
189, 190, 193, 194, 195, 198, 200, 202, 203, 204, 205, 208, 209, 211, 212, 213, 214,
216, 217, 218, 219, 220, 224, 231, 233, 240, 245, 246, 247, 251, 256, 265, 266, 267,
268, 270, 273, cccxvii, cccxxiii, cccxxiv, cccxxv

SDSS, 95, 109, 110, 111, 113, 130, 151, 152, 153, 157, 158, 161, 165

Secondary Eclipse, 16

sift, 121, 122, cccviii

Simple Storage Service, vi, 255

Sloan Digital Sky Survey, vi, vii, 2, 25, 38, 223

Source Catalogue, 197, 224, 225, 228, 241, 243

struct, 121, cccviii

Supernova, 202

syzygy, 23

Top-Down, 141, 142

Transit Method, 20, 60

Transit Time Variation, vi

Translation, 49, 56

Unix, v, vi, 44, 136, 147, 164

Variable Star, 13

Virtual Machine, vi, 49, 144, 305

Volunteer Computing, 42

wc, vi, 136, cccxii

wget, 54, 59, 105, 109

Worker Node, vi, 144, 170, 229, 253