

A Project Report on

AstroMiner:
Data Mining of Astronomical Databases

By

Jayant Gupchup
Sameer Pillai
Huzefa Rangwala
Mihir Shah
Devang Thakkar

Under the Guidance of

Dr. S. A. Patekar
Professor and Head
Dept. of Computer Technology
VJTI

&

Dr. A. K. Kembhavi
Dean (Visitor Academic Programmes)
IUCAA

Department of Computer Technology
Veermata Jijabai Technological Institute
Matunga, Mumbai - 400019

2002-2003

AstroMiner: Data Mining of Astronomical Databases

P R O J E C T

Submitted in partial fulfillment of the requirement
for the degree of

Bachelor of Engineering

By

Jayant Gupchup

Sameer Pillai

Huzefa Rangwala

Mihir Shah

Devang Thakkar

Under the guidance of

Dr. S. A. Patekar

Professor and Head

Dept. of Computer Technology

VJTI

&

Dr. A. K. Kembhavi

Dean (Visitor Academic Programmes)

IUCAA

Department of Computer Technology

Veermata Jijabai Technological Institute

University of Mumbai

2002-2003

CERTIFICATE OF APPROVAL

The project entitled:

AstroMiner:

Data Mining of Astronomical Databases

Submitted by:

Jayant Gupchup	GH-99417
Sameer Pillai	GH-99442
Huzefa Rangwala	GH-99444
Mihir Shah	GH-99449
Devang Thakkar	GH-99461

In partial fulfillment for the degree of Bachelor of Engineering in
Computer Engineering is approved.

Examiner
(Internal)

Examiner
(External)

C E R T I F I C A T E

This is to certify that the project titled

AstroMiner:

Data Mining of Astronomical Databases

is a bonafide record of the work done by

Jayant Gupchup	GH-99417
Sameer Pillai	GH-99442
Huzefa Rangwala	GH-99444
Mihir Shah	GH-99449
Devang Thakkar	GH-99461

during the year 2002-2003 under the joint guidance of:

Dr. S. A. Patekar, Veermata Jijabai Technological Institute (V.J.T.I.),
University of Mumbai.

&

Dr. A. K. Kembhavi, Inter-University Centre for Astronomy and Astrophysics
(IUCAA)

Head
Dept. of Computer
Technology

Dr. S. A. Patekar
Guide

Dr. A. K. Kembhavi
Guide

ACKNOWLEDGEMENTS

Our B.E. project has been a deeply enriching and gratifying experience. We would like to thank all the people who have helped us over the course of the past year – for the knowledge that we have gained and the invaluable exposure that the project has given us.

We are indebted to **Dr. Ajit Kembhavi**, Dean (Visitor Academic), IUCAA for his invaluable guidance and support. We would like to thank him for constantly challenging us and for giving us valuable insights from his expertise in the field of Astronomy. We would also like to thank him for giving us the opportunity to work at IUCAA and removing time to patiently listen to us on our trips to Pune.

We would like to thank **Ms. Sarah Ponrathnam**, in charge of computer systems at IUCAA for giving us all the facilities we needed to work on our project. We thank her for solving every technical difficulty we faced while working in the Computer Centre.

We would also like to thank **Ms. Pallavi Kulkarni**, who is working on the Virtual Observatory Project at IUCAA for solving our queries and doubts.

We would like to thank **Dr. S. A. Patekar**, Professor and Head, Department of Computer Technology, VJTI, for inspiring us to take up this project. We thank him for the belief he had in our work and us. We also thank him for all the support and guidance that he has provided us during the course of our project and over the years.

With sincere thanks,

Jayant Gupchup
Sameer Pillai
Huzefa Rangwala
Mihir Shah
Devang Thakkar

CONTENTS AT A GLANCE

1.	Introduction	1
2.	Astronomical Data	4
3.	Indexed Access of Astronomical Data	8
4.	Data Mining Techniques	17
5.	The WaveCluster Algorithm	25
6.	Cluster Analysis	41
7.	Visualisation	46
8.	User Interface	53
9.	Future Areas	59
	List of Illustrations	62
	List of Files	64
	List of References	65
	Appendix: Spectral Classes	67

TABLE OF CONTENTS

1.	Introduction	1
1.1	Statement of Problem	1
1.2	Scope of Problem	2
1.3	Requirements Analysis	2
2.	Astronomical Data	4
2.1	Astrometry	4
2.2	The Hipparcos Space Astrometry Mission	4
2.3	Astrometric Data	5
2.3.1	Right Ascension and Declination	5
2.3.2	Parallax	6
2.3.3	Proper Motion	6
2.4	Photometric Data	7
2.4.1	Visual Magnitude	7
2.4.2	Photographic Magnitude	7
2.4.3	Colour Index	7
3.	Indexed Access of Astronomical Data	8
3.1	Sequential Search	8
3.2	Indexed Search	8
3.2.1	k-d Trees	9
3.2.2	QuadTree	9
3.3	R-Tree	10
3.3.1	Searching in an R-Tree	12
3.3.2	Creation of an R-Tree	12
3.3.3	Node Splitting	13
3.3.4	Index on File	14
3.3.4.1	Searching in a File-Based R-Tree	15
3.3.4.2	Creating a File-Based R-Tree	15

4.	Data Mining Techniques	17
4.1	Knowledge Discovery in Databases	17
4.2	Data Mining Processes	20
4.3	Data Mining Algorithms	21
4.4	Clustering Algorithms	22
4.4.1	Requirements of a good clustering algorithm	22
4.4.2	Different Clustering Algorithms	23
5.	The WaveCluster Algorithm	25
5.1	Application of DSP to Spatial Databases	25
5.2	Motivation for Using Wavelet Transform	25
5.3	The Algorithm	26
5.3.1	Quantisation	26
5.3.2	Wavelet Transform	29
5.3.3	Connected-component Detection	31
5.3.3.1	Basic Pixel Connectivity	31
5.3.3.2	Connected Component Labelling Algorithm	31
5.3.3.3	Divide-and-Conquer Approach	32
5.3.4	Extraction of Cluster Components	36
5.4	Overview of WaveCluster Algorithm	38
6.	Cluster Analysis	41
6.1	The Hertzsprung-Russell (H-R) Diagram	41
6.1.1	Structure of H-R Diagram	42
6.1.2	Uses of H-R Diagram	43
6.2	Cluster Properties	45
7.	Visualisation	46
7.1	Equatorial Co-ordinate System	46
7.2	Galactic Co-ordinate System	51

8.	User Interface	53
8.1	GUI Design Model	54
8.2	Application Window	55
8.2.1	Visualisation Screen	55
8.2.2	Menu	56
8.2.3	Control Panel	58
9.	Future Areas	59
9.1	Using Larger Catalogues	59
9.2	Mining the Image Data	60
9.3	Mining on Other Parameters	61
9.4	Mining in Multiple Dimensions	61
	List of Illustrations	62
	List of Files	64
	List of References	65
	Appendix: Spectral Classes	67

CHAPTER ONE

Introduction

It is estimated that the total amount of information in the world doubles every 20 months.¹ This data explosion has meant that the size of databases as well as their numbers have increased dramatically. It is an increasing challenge to make efficient use of this vast amount of information.

As a result of this trend, the field of *Knowledge Discovery* has been developed to manage and manipulate this data in an efficient manner. *Data Mining* is a set of techniques that have been developed to extract useful information from this sea of knowledge. Data Mining has wide-ranging applications in a number of fields from the commercial to the academic spheres.

1.1 Statement of Problem

The study of celestial objects is called *Astronomy*. Modern methods involve a lot of information in the form of tabular databases as well as photographic data. These tabular databases are popularly called *catalogues*. Catalogues range from a few thousand to many million entries in size. While studying groups of stars, of particular interest are aggregates of stars occurring densely together. These clusters consist of groups of stars bound together by mutual gravitational attraction and detecting them required specialised techniques.

We are interested in detecting such clusters from astronomical databases. Once detected, we would like to analyse these clusters to study useful properties. The information that is extracted from these large databases should also be displayed to the user in a form that is clearly understandable. Thus we can clearly define the aim of our project.

Aim: To detect, analyse and visualise spatial clusters from astronomical databases.

¹ <http://bioinfo.weizmann.ac.il/cards/knowledge.html>

1.2 Scope of Problem

The scope of our project can be explained as follows:

- A large number of astronomical catalogues are available for analysis. However, we restrict the scope of our search to the *Tycho* and *Hipparcos*² catalogues.
- To allow us to access large astronomical databases quickly, we need to develop an index based search procedure. As each star is uniquely by its position in the sky, namely *Right Ascension* and *Declination*³, this index needs to be two-dimensional.
- To detect the clusters, we need to adopt a suitable technique. The databases are very large in size and hence this problem is a good candidate for the application of Data Mining techniques. We can use a suitable *Clustering algorithm*⁴ for the detection of spatial clusters. We make use of the *WaveCluster*⁵ algorithm in our project.
- To carry out the analyses of the detected clusters, we use the *Hertzprung-Russel*⁶ diagram. We further analyse the cluster by computing useful statistics.
- The results of the clustering process are provided to the user in a graphical format by
 - i. Plotting the cluster as it would appear in the sky.
 - ii. Displaying the cluster in terms of *Galactic Co-ordinates*⁷.

1.3 Requirements Analysis

The solution should satisfy the following requirements:

- The software should allow the user to specify a search window to carry out the mining in a particular area of interest.
- The user should be allowed to carry out the mining at different resolutions.
- The software should be efficient enough to carry out mining in real-time.

² Refer section 2.2 for detailed information

³ Refer section 2.3.1 for detailed information

⁴ Refer section 4.4 for detailed information

⁵ Refer chapter 5 for detailed information

⁶ Refer section 6.1 for detailed information

⁷ Refer section 7.2 for detailed information

- The results of the analysis should be presented such that the user is easily able to isolate the stars occurring in the cluster from the ones in the background.
- The visualisation should be quick enough to be executed whenever desired by the user.
- The analysis should be rigorous enough to allow the user to identify legitimate clusters from artificial ones.
- The user should be allowed to save legitimate clusters and later retrieve the mined data.

CHAPTER TWO

Astronomical Data

2.1 Astrometry

Astrometry is a part of Astronomy that deals with the positions of stars and other celestial bodies. It is one of the oldest areas of Astronomy, dating back to Hipparchus, who compiled the first catalogue of stars visible to him and in the process, he invented the brightness scale basically still in use today.⁸ The major objective of Astrometry is to provide a non-rotating stellar reference frame, which can be used to locate the position of all celestial bodies in the galaxy. Using this reference frame, we can obtain basic observational data for the study of stellar properties such as mass, luminosity, spatial distribution and their motions.

2.2 The Hipparcos Space Astrometry Mission

A large number of astronomical missions have been dedicated to the task of collecting accurate data about celestial objects. These missions collect celestial data by dividing the sky into a number of smaller portions, focusing on portions one by one and then merging the collected data. The data collected during these missions is available in the form of catalogues and helps astronomers in gaining in-depth knowledge about the objects to be studied. In our project we are focusing on the *Hipparcos* and the *Tycho* catalogues.

Named after the second century BC astronomer Hipparchus, Hipparcos or the **H**igh **P**recision **P**ARallax **C**ollecting **S**atellite was launched on 8th August 1989. The main instrument was designed to measure about one hundred thousand stars, the least intense of which were about 12 mag. The payload also included two star mappers, which were used by the Tycho experiment to perform astrometric and two-colour photometric measurements of about one million stars, the least intense of which were about 10-11 mag. The Hipparcos contains details of about 1,18,000 stars and contains 77 fields, and the Tycho catalogue contains details of 1 million stars specified using 57 fields. The basic fields of these catalogues specify the position of the star, its intensity, parallax etc. Other fields in the catalogues are for the purpose of easy identification and organisation.

⁸ To understand the magnitude scale, refer to Section 2.4. Lower magnitudes correspond to brighter stars.

2.3 Astrometric Data

For the purpose of our project we would be focusing on the positional information of the stars. Some fields pertaining to positional information are as follows.

2.3.1 Right Ascension and Declination

Astronomers have been using various reference systems for locating the positions of heavenly bodies. Most of the systems are based on the model of celestial sphere.

Celestial sphere is a sphere of infinite radius with observer on earth as center. The distant stars are projected over this sphere. One of the most popular methods uses *Right Ascension* (RA) and *Declination* (DEC) to specify the co-ordinates of stars. The Tycho contains the position of about one million stars as described by their RA and DEC.

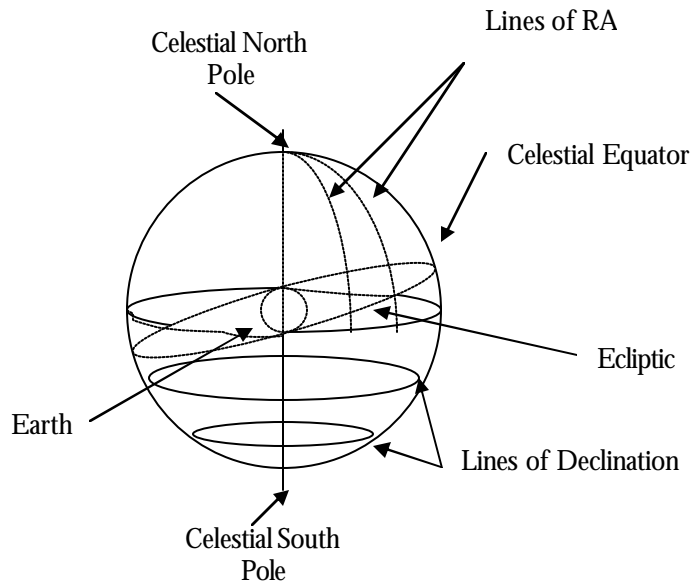


Figure 2.1 RA and DEC

Celestial equator is a circle where the celestial sphere cuts the earth's equatorial plane. The lines of Declination are as shown, parallel to celestial equator in figure 2.1. DEC is measured in Degree: Minute: Second. The declination of celestial North Pole is $+90^\circ$ and that of celestial South Pole is -90° .

Figure 2.1 also explains the Right Ascension. The reference used for measurement of RA is the great vertical circle that passes through vernal equinox.

Vernal equinox is the point where ecliptic cuts celestial sphere on March 21st. *Ecliptic* is the path of the sun across celestial sphere. RA is measured in Hour: Minute: Second.

2.3.2 Parallax

There is an apparent shift in the position of the stars when viewed from two widely separated points. Due to revolution of earth around the sun, this apparent shift is observed. This shift, called *Parallax*, can be measured as angle in mas (milli arc second). Accurate distances of stars can be calculated if parallax is measured accurately. Due to parallax, the position of star x appears to be in the direction of a at one position and in direction b at another. Thus, parallax provides information about a star's distance from the Earth.

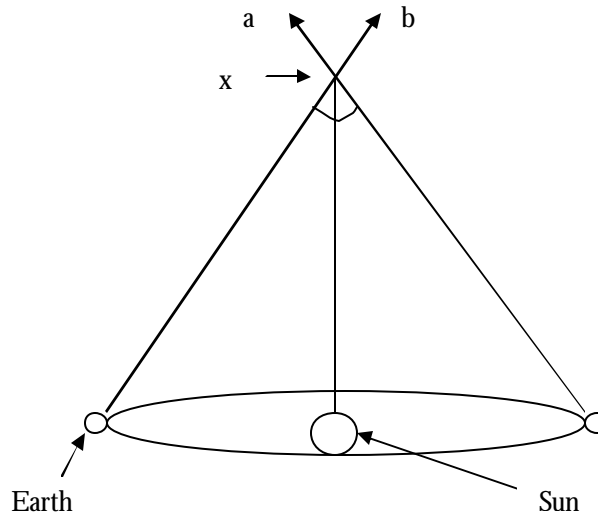


Figure 2.2 Parallax

2.3.3 Proper Motion

Apart from the motion due to parallax, some stars actually move in particular directions. This motion of stars is called *Proper Motion*. As the star moves from A to B it moves through angle μ . The angle through which a star appears to move in one year is called the Proper Motion of star. Proper Motion can be resolved in radial and tangential components. This is measured in mas/yr. Thus RA (proper motion) and Dec (proper motion) specify proper motion.

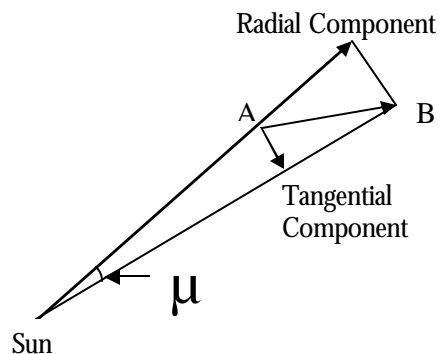


Figure 2.3 Proper Motion

2.4 Photometric Data

Apart from the positional or astrometric data the catalogues also contain important data pertaining to the brightness, intensity and colour of the celestial object. This data is also known as photometric data. Some important fields of photometric data are as follows.

2.4.1 Visual Magnitude

The *Visual or Apparent Magnitude* of a star approximates to the brightness of the star as seen by the unaided eye. This magnitude may be designated as V for visual. An important feature of this is that the brighter the star less is its visual magnitude. In fact, an increase in the brightness of a star by a factor of 2.5 corresponds to a decrease of visual magnitude by 1. To obtain this magnitude the brightness of the star is measured through a yellow colour filter. The reason yellow is chosen is that it is at the center of the visible spectrum.

2.4.2 Photographic Magnitude

Currently photometry is carried out with electronic instruments, but in earlier years photometry was carried out by measuring the exposure produced by the image of a star on a photographic plate. In comparison with the response of the human eye, a photographic emulsion is relatively more sensitive to short wavelength blue light than it is to red. Thus a blue star would produce more exposure on a photographic emulsion than would be expected from its visual magnitude, while a red star would produce less exposure. Therefore, the unfiltered photographic image of a star gives a measure of the star's magnitude weighted towards the blue end of the spectrum. This is the B (for blue) or *Photographic Magnitude*. (Optical and electronic filtering allow modern electronic detectors to approximate the colour response of the eye or of a photographic emulsion.)

2.4.3 Colour Index

The difference $B - V$ between the two magnitude estimates is known as the *B- V Colour Index* of the star. It gives a numerical measurement of the colour of a star. For blue stars it will be negative, while for very red stars, it will be a positive number.

CHAPTER THREE

Indexed Access of Astronomical Data

The Hipparcos catalogue contains around 118,000 records while the Tycho catalogue contains around 1,000,000 records. As a result, quick and efficient access of the records plays an important role in any application based on these catalogues. For the purpose of detecting star clusters, we require to find stars that lie within a given range of *Right Ascension* (RA) and *Declination* (DEC). Thus, the database can be considered to be a spatial (two-dimensional) database with RA and DEC as the key fields.

3.1 Sequential Search

The simplest form of search, the *Sequential search* would be highly inefficient as it would take on an average $n/2$ comparisons to search in a file containing n records. In our case, it would take on average 60,000 and 500,000 comparisons respectively to search in the Hipparcos and Tycho catalogues.

3.2 Indexed Search

An index based on the spatial nature of the database is desirable. Classical one-dimensional index structures such as *B-Trees* will not work since the search space is two-dimensional. Structures based on exact matching of values, such as *Hash Tables* are not useful because a range search is required. Two-dimensional index structures are ideally suited to our applications. We consider the following structures.

- k-d Tree
- QuadTree
- R-Tree

3.2.1 k-d Trees

Each level of a *k-d Tree* partitions the space into two in such a way that at each node, approximately one-half of the points stored in the sub-tree fall on one side, and one-half on the other. The partitioning is done along one dimension at the node at the top level of the tree, along the second dimension in nodes at the next level and so on, cycling through the dimensions. Partitioning stops when a node has less than a given maximum number of points.

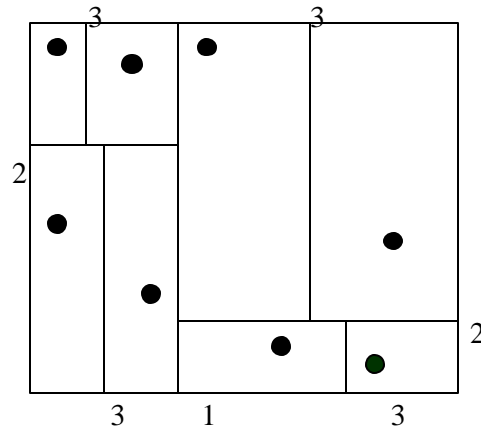


Figure 3.1 k-d Tree

3.2.2 QuadTree

Each node of a *Quadtree* is associated with a rectangular region of space. The top node is associated with the entire sample space. Each non-leaf node in a Quadtree divides its region into four equal-sized quadrants corresponding to four child nodes. Leaf nodes have between zero and some fixed maximum number of points. Hence, if a region corresponding to a node has more than the maximum number of points, child nodes are created for that node.

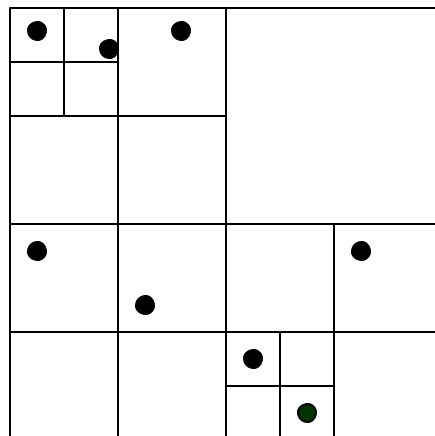


Figure 3.2 QuadTree

The disadvantages of using k-d trees and Quadrees for our application are as follows:

- Many of the entries within a node are wasted because of the rigid rules for the splitting of a node. As a consequence, the memory space required to store the tree is more.
- The height of the tree may be arbitrarily large. As a result, searching requires more number of steps.

To overcome these practical difficulties, we use a more dynamic indexing structure called R-Trees that represents data objects by intervals in more than one dimension.

3.3 R-Tree

An *R-Tree* is a height-balanced tree similar to a B-Tree with index records in its leaf nodes containing pointers to data objects. The Hipparcos and Tycho catalogues consist of a collection of records representing stars. Each record has an *Offset-value* that gives the position of the record within the catalogue file.

Leaf nodes in the R-Tree contain index record entries of the form

$$(I, \text{Offset-value})$$

where I is the two-dimensional position identifier of the star indexed. Therefore,

$$I = (RA, DEC)$$

Non-leaf nodes contain entries of the form

$$(I, \text{Child-Pointer})$$

where *Child-Pointer* is the address of a lower node in the R-Tree and I is the smallest rectangle that covers all the rectangles in the lower node's entries. Therefore,

$$I = (RA_{min}, DEC_{min}, RA_{max}, DEC_{max})$$

Let M be the maximum number of entries that will fit in one node and let $m \leq M/2$ be a parameter specifying the minimum number of entries in a node. Hence, each node (except the root) contains between m and M number of entries. The root node has at least two children unless it is a leaf.

The height of an R-Tree containing N index records is at most $\lceil \log_m N \rceil - 1$. For our application we have selected $M = 20$ and $m = 10$. With these values of m and M , the maximum height of the tree for both the Hipparcos catalogue ($N = 100,000$) and the Tycho catalogue ($N = 1,000,000$) is just 6.

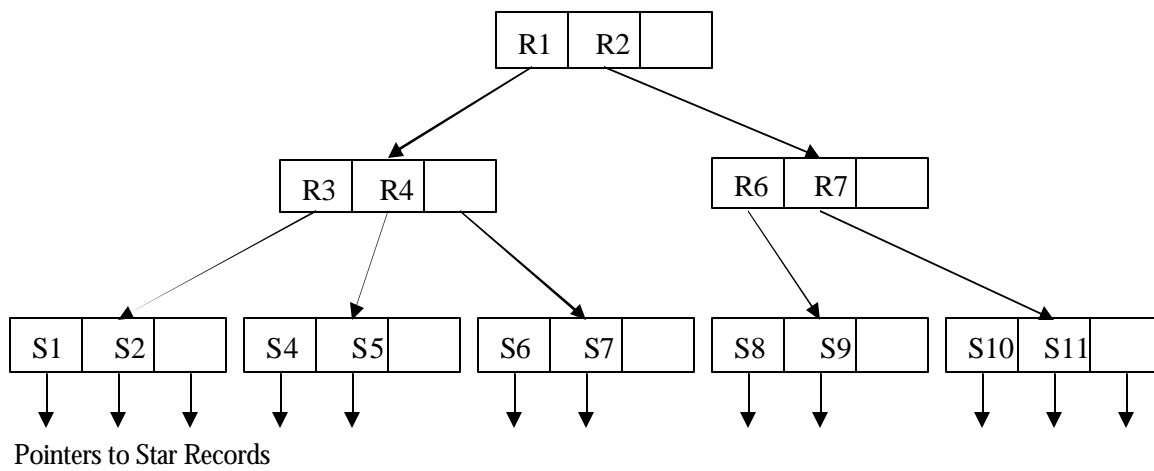
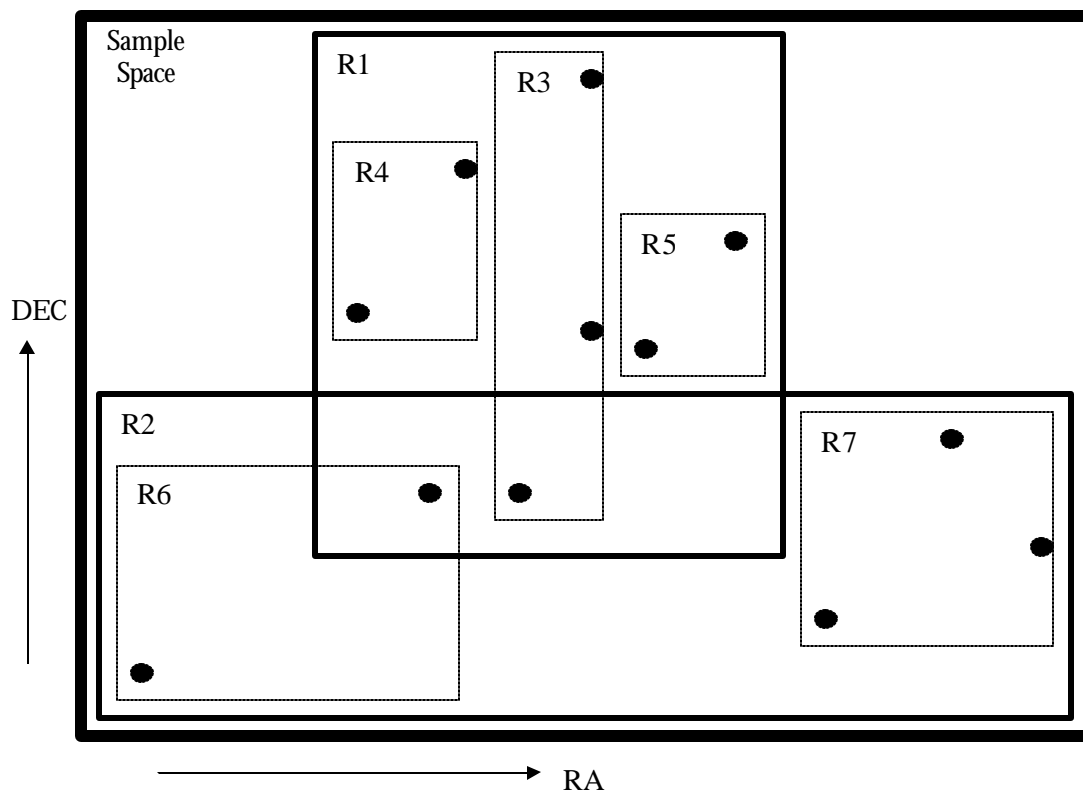


Figure 3.3 Sample space and the corresponding R-Tree.

We have taken $M = 3$ and $m = 2$



In the algorithms that follow, we denote the rectangle part of an index entry E by EI and the *Offset-value* or *Child-Pointer* by Ep .

3.3.1 Searching in an R-Tree

The search algorithm descends the tree from the root. However, more than one sub-tree under a node visited may need to be searched, hence it is not possible to guarantee good worst-case performance. Generally, the tree is maintained in a form that allows the search algorithm to eliminate irrelevant regions of the indexed space and examine only data near the search area. Along any one path, the minimum and maximum number of checks is equal to $\lceil \log_m N \rceil$ and $\lceil \log_m N \rceil * M$ respectively. With respect to the catalogues we are using, this gives a minimum and maximum value of 6 and 120 respectively. This number is very small in comparison with the number of checks required for *Sequential* search.

- *Algorithm Search*
Given an R-Tree whose root node is T , find all index records whose rectangles overlap a search rectangle S .
 1. If T is not a leaf, check each entry E to determine whether EI overlaps S . For all overlapping entries, invoke *Search* on the tree whose root node is pointed to by Ep .
 2. If T is a leaf, check all entries E to determine whether EI overlaps S . If so, E is a qualifying record.

3.3.2 Creation of an R-Tree

During the insertion of index records for new data records into an R-Tree, the new index records are added to the leaf nodes, nodes that overflow are split, and splits propagate up the tree.

- *Algorithm Insert*
Insert a new index entry E into an R-Tree
 1. Invoke *ChooseLeaf* to select a leaf node L in which to place E .
 2. If L has room for another entry, install E . Otherwise invoke *SplitNode* to obtain two leaf nodes L and LL containing E and all the old entries of L .
 3. Invoke *AdjustTree* on L , also passing LL if a split was performed
 4. If node split propagation caused the root to split, create a new root whose children are the two resulting nodes.

- Algorithm *ChooseLeaf*

Select a leaf node in which to place a new index entry E .

1. Set N to be the root node.
2. If N is a leaf, return N .
3. If N is not a leaf, let F be the entry in N whose rectangle EI needs the least enlargement to include EI . Resolve ties by choosing the entry with the rectangle of smallest area.
4. Set N to be the child node pointed to by Ep and repeat from Step 2.

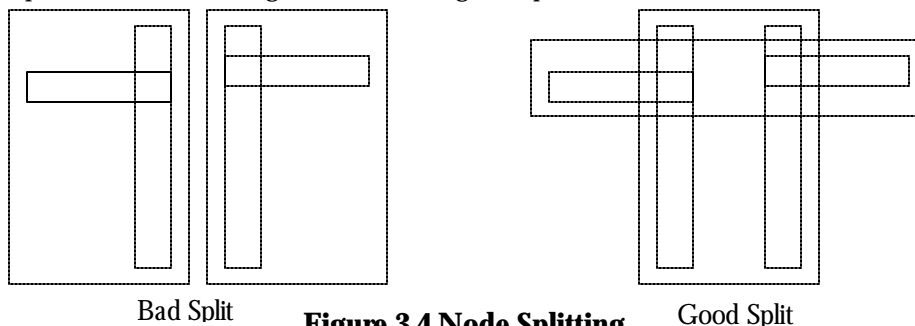
- Algorithm *AdjustTree*

Ascend from a leaf node L to the root, adjusting covering rectangles and propagating node splits as necessary.

1. Set $N = L$. If L was split previously, set NN to be the resulting second node.
2. If N is the root, stop.
3. Let P be the parent node of N , and let E_N be N 's entry in P . Adjust $E_N I$ so that it tightly encloses all entry rectangles in N .
4. If N has a partner NN resulting from an earlier split, create a new entry E_{NN} with $E_{NN}p$ pointing to NN and $E_{NN}I$ enclosing all rectangles in NN . Add E_{NN} to P if there is room. Otherwise, invoke *SplitNode* to produce P and PP containing E_{NN} and all P 's old entries.
5. Set $N = P$ and set $NN = PP$ if a split occurred. Repeat from Step 2.

3.3.3 Node Splitting

In order to add a new entry to a full node containing M entries, it is necessary to divide the collection of $M + 1$ entries between two nodes. The division is done in a way that makes it as unlikely as possible that both new nodes will need to be examined on subsequent searches. Since, the decision whether to visit a node depends on whether its covering rectangle overlaps the search area, the total area of the two covering rectangles should be minimized. As shown in figure 3.4, the area of the covering rectangles in the “bad split” case is much larger than in the “good split” case.



- Algorithm *SplitNode* –
Divide a set of $M + 1$ index entries into two groups.
 1. Apply algorithm *PickSeeds* to choose two entries to be the first elements of the two groups. Assign each to a group.
 2. If all entries have been assigned, stop. If one group has so few entries that all the rest must be assigned to it in order for it to have the minimum number m , assign them and stop.
 3. Invoke algorithm *PickNext* to choose the next entry to assign. Add it to the group whose covering rectangle will have to be enlarged the least to accommodate it. Resolve ties by adding the entry to the group with smaller entries and then to one with fewer entries. Repeat from Step 2.
- Algorithm *PickSeeds* –
Select two entries to be the first elements of the groups.
 1. For each pair of entries E_1 and E_2 , compose a rectangle J including E_1I and E_2I . Calculate $d = \text{area}(J) - \text{area}(E_1I) - \text{area}(E_2I)$.
 2. Choose the pair with the largest d .
- Algorithm *PickNext* –
Select one remaining entry for classification in a group.
 1. For each entry E not yet in a group, calculate $d_1 = \text{area increase required in the covering rectangle of Group 1 to include } EI$. Similarly, calculate d_2 for Group 2.
 2. Choose any entry with the maximum difference between d_1 and d_2 .

3.3.4 Index on File

In our application we will be dealing with databases that have entries of the order of a few million. Creating an index for such a database will be highly time consuming. The efficient solution is to create the index just once and store it on file. We use this index every time we run the mining algorithm.

Also, indexing such databases results in indices that are several megabytes in size. Loading the entire index in memory every time we run the mining algorithm is a waste of valuable physical memory. For very large databases indices may even be larger than total physical memory. Our application will read only one entry at a time from the index on the file. Thus we need a minimum of physical memory and we have very fast access of the database using very few file accesses.

The Tycho catalogue contains 1058332 entries. The catalogue is about 354 MB in size. The index created using 20 entries per node has a height of 6 and is 36.5 MB in size. Hence we can search for any entry using 6 file accesses and any path from root to leaf will result in a minimum of 6 comparisons and a maximum of 120.

In the algorithms that follow, we denote the rectangle part of an index entry E by EI and the *Offset-value* or *Child-Pointer* by Ep . For a non-leaf node, Ep gives the offset of the child record in the index file. For a leaf node, it gives the record number in the main database.

3.3.4.1 Searching in a File-Based R-Tree

- Algorithm *FileIndexSearch*
Given an R-Tree stored on file, find all index records whose rectangles overlap a search rectangle S and create a *SearchedFile* containing their corresponding record numbers in the main database. Let *Current-Offset* be 0.
 1. Read the record T stored at *Current-Offset*.
 2. If T is not a leaf, check each entry E to determine whether EI overlaps S . For all overlapping entries, invoke *FileIndexSearch* with *Current-Offset* set to Ep .
 3. If T is a leaf, check all entries E to determine whether EI overlaps S . If so, E is a qualifying record and store Ep in *SearchedFile*.

3.3.4.2 Creating a File-Based R-Tree

Let *NodeCount* be initialised to 0. Create an empty record at the start of the file.

- Algorithm *FileIndexInsert*
Insert a new index entry E into an R-Tree stored on file
 1. Invoke *ChooseLeaf* to select a leaf node record L in which to place E .
 2. If L has room for another entry, install E in L and overwrite the old L with the updated L . Otherwise invoke *SplitNode* to obtain two leaf node records L and LL containing E and all the old entries of L . Overwrite the old L with the updated L . Write LL at the end of the file. Increment *NodeCount* by 1.
 3. Invoke *AdjustTree* on L , also passing LL if a split was performed.
 4. If node split propagation caused the root to split, create a new root whose children are the two resulting nodes. Write the root at the end of the file. Increment *NodeCount* by 1.

- Algorithm *FileIndexChooseLeaf*
Select a leaf node in which to place a new index entry E .
 1. Set *Current-Offset* to the offset of the root node record.
 2. Read the record N stored at *Current-Offset*.
 3. If N is a leaf, return N .
 4. If N is not a leaf, let F be the entry in N whose rectangle FI needs the least enlargement to include EI . Resolve ties by choosing the entry with the rectangle of smallest area.
 5. Set *Current-Offset* to Ep and repeat from Step 2.
- Algorithm *FileIndexAdjustTree*
Ascend from a leaf node record L to the root, adjusting covering rectangles and propagating node splits as necessary.
 1. Set $N = L$. If L was split previously, set NN to be the resulting second node.
 2. If N is the root, stop.
 3. Let P be the parent node record of N , and let E_N be N 's entry in P . Adjust $E_N I$ so that it tightly encloses all entry rectangles in N .
 4. If N has a partner NN resulting from an earlier split, create a new entry E_{NN} with $E_{NN}p$ pointing to NN and $E_{NN}I$ enclosing all rectangles in NN . Add E_{NN} to P if there is room. Otherwise, invoke *SplitNode* to produce P and PP containing E_{NN} and all P 's old entries.
 5. Overwrite the old P with the updated P .
 6. If a split occurred, write PP at the end of the file. Increment *NodeCount* by 1.
 7. Set $N = P$ and set $NN = PP$ if a split occurred. Repeat from Step 2.

At the end of inserting all the records, copy the index record corresponding to the root at the start of the index file.

CHAPTER FOUR

Data Mining Techniques

Data Mining can be defined as follows:

Def. 4.1: Data Mining. *Data mining* is the exploration and analysis, by automatic or semiautomatic means, of large quantities of data in order to discover meaningful patterns and rules.

These meaningful patterns are used to improve business practices including marketing, sales, and customer management. The finding of useful patterns in data has been referred to as knowledge extraction, information discovery, information harvesting, data archaeology, and data pattern processing in addition to data mining. In recent years the field has settled on data mining to describe these activities.

4.1 Knowledge Discovery in Databases

Statisticians have commonly used the term data mining to refer to the patterns in data that are discovered through multivariate regression analyses and other statistical techniques. However, as data mining has matured, it is widely accepted to be a single phase in a larger process known as Knowledge Discovery in Databases or KDD for short. The field of KDD is particularly focused on the activities leading up to the actual data analysis and includes the evaluation and deployment of results. Following are the rudimentary steps involved in KDD.

- *Data Selection* – The goal of this phase is the extraction of data, relevant to the data mining process, from a large store. This data extraction helps to streamline and speed up the process.
- *Data Preprocessing* – This phase of KDD is concerned with data cleansing and preparation tasks that are necessary to ensure correct results. Eliminating missing values in the data, ensuring that coded values have a uniform meaning and ensuring that no spurious data values exist, are typical actions that occur during this phase.
- *Data Transformation* – This phase of the lifecycle is aimed at converting the data into a two-dimensional table and eliminating unwanted fields so the results are valid.

- *Data Mining* – The goal of the data mining phase is to analyse the data by an appropriate set of algorithms in order to discover meaningful patterns and rules and produce predictive models. This is the core element of the KDD cycle.
- *Interpretation and Evaluation* – While data mining algorithms have the potential to produce an unlimited number of patterns hidden in the data, many of these may not be meaningful or useful. This final phase is aimed at selecting those models that are valid and useful for making future business decisions.

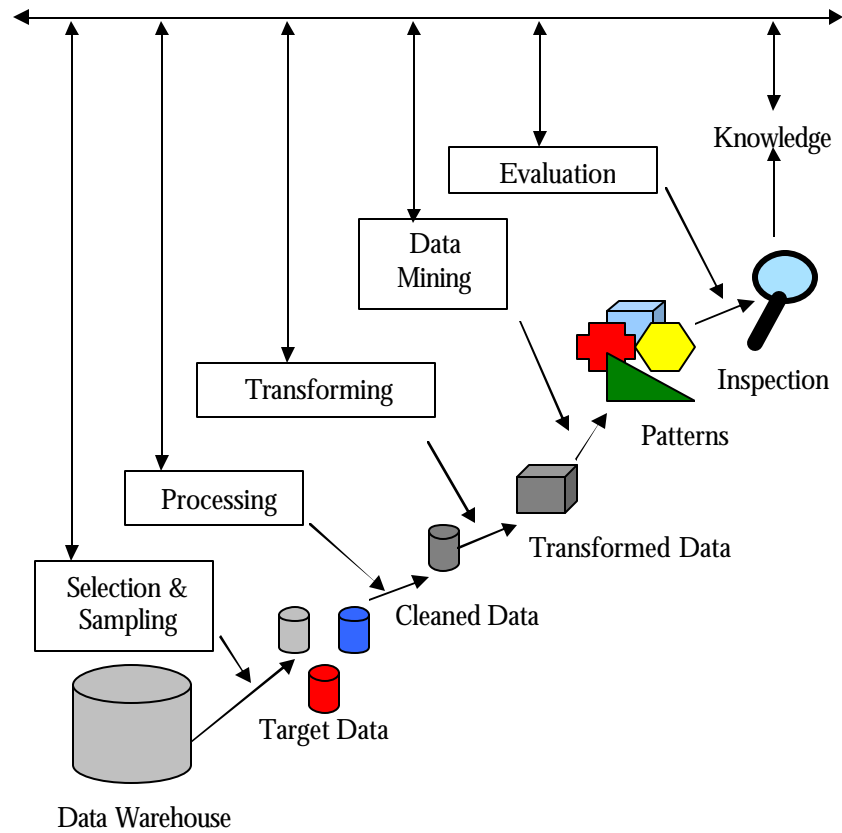


Figure 4.1 The Traditional KDD Paradigm

The KDD process in our project is explained as follows:

The data that is to be used in our project consists of astronomical databases, viz. the Hipparcos and Tycho catalogues. These catalogues are plaintext files, where the records are stored in a continuous manner. Thus, though the data represented in the file is tabular or two-dimensional data, it is stored on the file in a one-dimensional format. Each record is given a unique *Record ID* to uniquely identify it and the database is sorted on this Record ID.

As we are using these databases where the data extraction has already taken place, the phase of Data Selection is eliminated. These databases have high integrity. There are no duplicate values and the unique identifiers for every record, viz. the Record ID and the combination of RA and DEC. Hence the phase of Preprocessing can also be eliminated.

Knowing the Record ID, we need to develop a system to access that record directly. In other words, we need to be able to access this one-dimensional data as if it were a two-dimensional table. This is the phase of Data Transformation. This can be done as every record is identical in size and the primary key of the database is the Record ID.

Suppose we want to access the record whose Record ID is x . If the size of each record is y bytes in length, then the *offset* that will give us the start of record x can be calculated as follows:

$$\text{offset} = (x - 1) * y$$

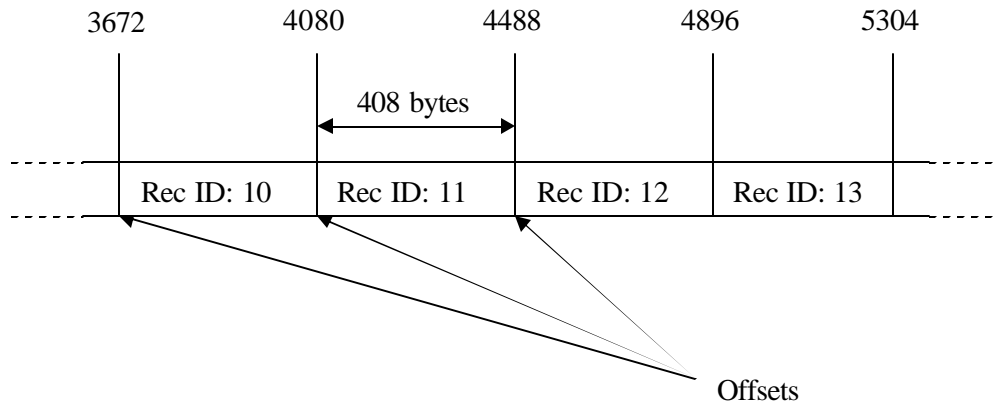


Figure 4.2 Organisation of Records in File

Thus, we can directly go to our record of choice and we can access the database as if it were in the tabular format.

The phases of Data Mining and Interpretation can now be applied to this transformed data to allow the extraction of useful data, viz. spatial clusters of stars and the KDD process can be completed.

4.2 Data Mining Processes

Traditionally, there have been two types of statistical analyses - *Confirmatory analysis* and *exploratory analysis*. In confirmatory analysis, one has a hypothesis and either confirms or refutes it. However, the bottleneck for confirmatory analysis is the shortage of hypotheses on the part of the analyst. In "exploratory analysis", one finds suitable hypotheses to confirm or refute. Here the system takes the initiative in data analysis, not the user. From a process-oriented view, there are three classes of data mining activity: *discovery*, *predictive modeling* and *forensic analysis* as shown in Figure 4.3.

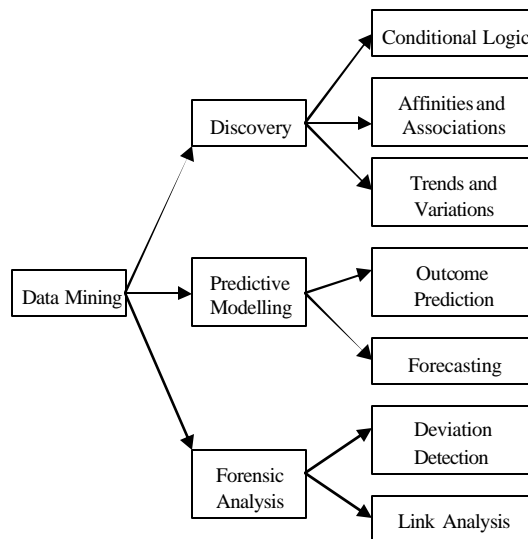


Figure 4.3 Processes in Data Mining

Discovery is the process of looking in a database to find hidden patterns without a predetermined idea or hypothesis about what the patterns may be. In other words, the program takes the initiative in finding what the interesting patterns are, without the user thinking of the relevant questions first. In large databases, there are so many patterns that the user can never practically think of the right questions to ask. The usefulness of this technique is determined by the richness and transparency of the patterns discovered and the quality of the inferred information.

In *Predictive Modelling* patterns discovered from the database are used to predict the future. Predictive modelling thus allows the user to submit records with some unknown field values, and the system will guess the unknown values based on previous patterns discovered from the database. While discovery finds patterns in data, predictive modelling applies the patterns to guess values for new data items.

Forensic analysis is the process of applying the extracted patterns to find anomalous or unusual data elements. To discover the unusual, we first find what is the norm, and then we detect those items that deviate from the usual within a given threshold. Note that discovery helps us find "usual knowledge," but forensic analysis looks for unusual and specific cases.

4.3 Data Mining Algorithms

The following are summaries of some of the industry-accepted algorithm types for automated data mining.

- *Classification Algorithms and Decision Trees* – Determines natural splits in the data based on a target variable. First splits occur on the most significant variables. A branch in a decision tree can be viewed as the conditional side of a rule.
- *Rule Association* – Identifies cause and effect relationships and assigns probabilities or certainty factors to support the conclusions. Rules are of the form "if <condition>, then <conclusion>" and can be used to make predictions or estimate unknown values.
- *Memory-based Reasoning (MBR) or Case-based Reasoning (CBR)* – These algorithms find the closest past analogs to a present situation in order to estimate an unknown value or predict an unknown outcome.
- *Cluster Analysis* – Separates heterogeneous data into homogeneous and semi-homogeneous subgroups based on the assumption that observations tend to be like their neighbours. Clustering increases the ability to make predictions.
- *Artificial Neural Networks* – Uses a collection of input variables, mathematical activation functions, and weightings of inputs to predict the value of target variables. Through an iterative training cycle, a neural network modifies its weights until the predicted output matches actual values. Once trained, the network is a model that can be used against new data for predictive purposes.
- *Genetic Algorithms* – Uses a highly iterative process of selection, crossover, and mutation operations to evolve successive generations of models. A fitness function is used to keep certain members and discard others. Genetic algorithms are primarily used to optimise neural network topologies and weights. However, they can be used by themselves for modeling.

4.4 Clustering Algorithms

The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering. A cluster is thus a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in another cluster. Clustering thus helps in separating a given dataset into smaller classes and then carrying out useful analysis of each class.

4.4.1 Requirements of a good clustering algorithm

The requirements of a good clustering algorithm are listed as follows:

- *Scalability* – It should be able to deal with data sets containing a large number of objects. It should not be affected by the size of the database.
- *Ability to deal with different types of attributes* – It should not be restricted by the type of attribute used for the purpose of clustering. For example some algorithms are designed to cluster numerical data, while certain applications may require other types of data, such as binary, categorical, etc.
- *Discovery of clusters with arbitrary shape* – Many clustering algorithms mine clusters based on fixed distance measures and hence tend to find spherical clusters with similar size and density. However, a cluster could be of any shape and hence a good clustering algorithm should be able to detect clusters of any arbitrary shape.
- *Minimal requirements of domain knowledge to determine input parameters* – Clustering algorithms sometimes tend to depend on input parameters such as number of clusters in order to carry out the clustering process. However, such data is hard to determine and hence a good clustering algorithm should be free from such a requirement.
- *Ability to deal with noisy data* – Most real-world databases contain outliers or missing, unknown, or erroneous data. A good clustering algorithm should not be sensitive to such data.
- *Insensitivity to the order of input records* – In some clustering algorithms the output of the algorithm depends upon the order in which the input records are presented to the algorithm. A good clustering algorithm should be insensitive to the order of input data.

- *High dimensionality* – A good clustering algorithm should be able to deal with large number of attribute or dimensions present in the database or data warehouse.
- *Interpretability and usability* – The clustering algorithm should be such that it is easy to adapt it with a particular user application and hence should be highly interpretable, comprehensible, and usable.

4.4.2 Different Clustering Algorithms

The detection of clusters can be achieved by employing any suitable clustering algorithm. The different clustering algorithms can be classified as follows:

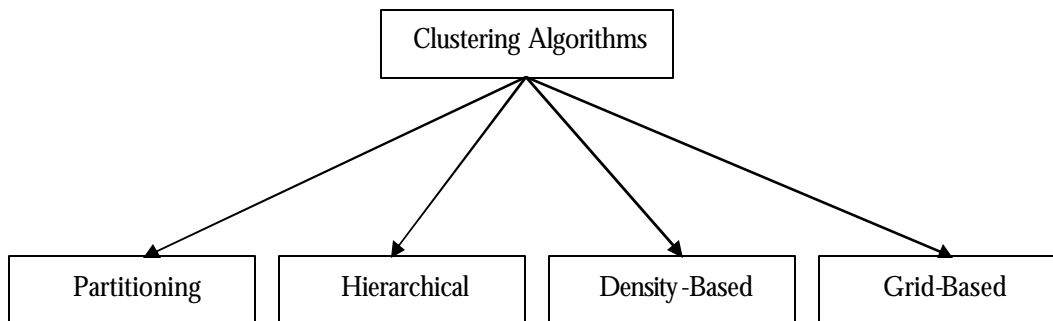


Figure 4.4 Clustering Algorithms

- *Partitioning algorithms* – Partitioning algorithms construct a partition of a database of N objects into a set of K clusters. Usually they start with an initial partition and then use an iterative control strategy to optimise an objective function. There are mainly two approaches, viz. k -means algorithm, where each cluster is represented by the center of gravity of the cluster and k -medoid algorithm, where each cluster is represented by one of the objects of the cluster located near the center. E.g. PAMS, CLARA, CLARANS
- *Hierarchical algorithms* – Hierarchical algorithms create a hierarchical decomposition of the database. The algorithm iteratively splits the database into smaller subsets until some termination condition is satisfied. Hierarchical algorithms do not need K as an input parameter, which is an obvious advantage over partitioning algorithms. The disadvantage is that the termination condition has to be specified. E.g. BIRCH

- *Density based algorithms* – They are unsupervised clustering algorithms to locate clusters by constructing a density function that reflects the spatial distribution of the data points. This method can find arbitrary shape clusters and does not make any assumptions about the underlying data distribution. This method is computationally very expensive and so can be impractical for very large databases. E.g. DBSCAN algorithms create a hierarchical decomposition of the database.
- *Grid-based algorithms* – Recently a number of algorithms have been presented which quantise the space into a finite number of cells and then do all operations on the quantised space. The main characteristic of these approaches is their fast processing time, which is typically independent of the number of data objects. They depend only on the number of cells in each dimension in the quantised space. E.g. WaveCluster

For our project, we propose WaveCluster, a grid-based approach for a number of reasons. The proposed approach is very efficient, especially for very large databases. The computational complexity of generating clusters in our method is $O(N)$. The results are not affected by outliers and the method is not sensitive to the order of the number of input objects to be processed. WaveCluster is capable of finding arbitrary shape clusters with complex structures such as concave or nested clusters at different scales, and does not assume any specific shape for the clusters. A-priori knowledge about the exact number of clusters is not required in WaveCluster, however, an estimation of expected number of clusters helps in choosing the appropriate resolution of clusters. WaveCluster is also multi-resolution capable, i.e. the grid size can be adjusted to extract clusters at any desired resolution. We discuss this algorithm in detail in the next chapter.

CHAPTER FIVE

The WaveCluster Algorithm

The WaveCluster Algorithm is a clustering technique based on the wavelet transform. The wavelet transform is a signal processing technique and can also be applied to spatial databases such as astronomical catalogues.

5.1 Application of DSP to Spatial Databases

The primary reason for application of signal processing techniques to spatial databases is that objects from the database can be represented in an n -dimensional feature space. Every object is associated by a number of numerical attributes. Thus, with every object we can associate a feature vector whose elements consist of these numerical attributes. With the help of these feature vectors we can represent each of these objects in a multidimensional spatial area called the feature space. Each element of the feature vector corresponds to one dimension of the feature space. The distribution of the objects is generally not uniform and we can use signal-processing techniques to identify the dense clusters and hence identify the overall distribution of the objects.

5.2 Motivation for Using Wavelet Transform

The motivation for using the wavelet transform is based on the following observations:

- *Unsupervised clustering* – The shape of the mother wavelet ensures that in a cluster, the points that lie within the cluster are emphasised, while at the same the points that lie on the boundaries of the cluster are suppressed. Thus we see that the dense regions act as attractors for points that lie within the cluster and inhibitors for points that lie just outside. Thus we see that the clusters automatically clear out the region around them, making themselves more distinct.
- *Effective removal of outliers* – The use of low-pass filters to carry out the transform ensures that outliers are removed once the wavelet transform is applied.
- *Multi-resolution property* – As we are using a signal-processing technique, we can use the inherently present multi-resolution property to detect the clusters at different levels of accuracy. The clusters can be detected at different resolutions from coarse to fine.

- *Cost-efficiency* – The wavelet transform is very fast and it makes our approach very cost-effective. Detecting clusters takes only a few seconds, which makes it possible to carry out the mining in real time.

5.3 The Algorithm

We consider the multidimensional spatial data, i.e. the feature space, to be a multidimensional input signal. When there are a large number of objects, we can apply signal-processing techniques to detect the clusters. This is done by using techniques such as the wavelet transform to convert the input to the frequency domain. The high frequency portions of the signal consist of those parts where there is a rapid change in the distribution of objects, i.e. the boundaries of the clusters. The low frequency, high-amplitude portion of the signal consists of the actual clusters themselves. The transformation is done by the convolution of the input data with an appropriate kernel function. This transformed feature space is free from outliers and the clusters are more distinguishable. We then apply a connected-components detection algorithm to identify the individual clusters.

Thus, the steps in the algorithm are:

1. Quantisation
2. Wavelet transform
3. Connected-component detection
4. Extraction of cluster components

We will now elaborate on the algorithm.

5.3.1 Quantisation

The first step in the algorithm is to convert the spatial data into a discrete multidimensional input signal. In our case the spatial data is two-dimensional, i.e. the catalogue of stars, where each star is identified uniquely by its RA and DEC. To convert the spatial data, we create a bitmap representing the area of the sky, which is of interest. Every element of the bitmap will correspond to a cell of finite area. The dimensions of this cell are specified by the user and each cell holds the count of the number of stars that lie inside the portion of the sky corresponding to the cell. Thus, to create the bitmap, we need to know the number of cells into which the entire area is to be divided.

$$m = (\text{RAmax} - \text{RAmin}) / \text{GridsizeX}$$

$$n = (\text{DECmax} - \text{DECmin}) / \text{GridsizeY}$$

Thus, if the RA dimension is divided into m intervals and the DEC dimension is divided into n intervals, the entire feature space will be divided into mn cells. Each star will be assigned to a cell, depending upon its position in the sky, i.e. depending upon its RA and DEC.

The bitmap can be treated as a matrix, where each element of the matrix is an individual cell. Hence, a cell can be reference by its row and column. A cell at position (x, y) will hold a star ($RA = r, DEC = d$) if:

$$GridsizeX * x \leq r \leq GridsizeX * (x + 1)$$

and

$$GridsizeY * y \leq d \leq GridsizeY * (y + 1)$$

- Algorithm *Quantisation*

Let RA_{min} , RA_{max} , DEC_{min} , DEC_{max} specify the boundaries of the search area specified by the user. Let $GridsizeX$, $GridsizeY$ specify the grid size in the RA and DEC direction respectively. Let *GridMatrix* be output $m * n$ two-dimensional matrix which is initialised to all 0's.

1. Use the R-Tree Index based search procedure *FileIndexSearch* to find all the stars that lie within the specified area and create an output file *SearchedFile* containing the record numbers of all these stars in the main database.
2. Read a record number from *SearchedFile*
3. Read the corresponding star record from the main database.
4. Let RA and DEC be the values of Right Ascension and Declination of the corresponding star.
5. $GridX = (RA - RA_{min}) / GridsizeX$
6. $GridY = (DEC - DEC_{min}) / GridsizeY$
7. $GridMatrix [GridX][GridY] = GridMatrix [GridX][GridY] + 1$

The output *GridMatrix* is stored as a GridFile called *QuantisedFile*, which is a representation of a two-dimensional matrix stored on file.

Files are stored in memory as one-dimensional flat text files. We would like to store a matrix of numbers in memory. Hence we need to develop a method to map the entire in the one-dimensional file to the elements of the two-dimensional matrix. We have developed the GridFile to enable us to do just this.

Let us assume that the dimensions of the matrix are $m \times n$. To translate the two-dimensional data into a one-dimensional format, we store the rows sequentially on the file. Thus, the elements of row 1 are stored, followed by the elements of row 2, and so on. Knowing the row and column of an individual element of the matrix, we can calculate its offset on the file and access it directly. The formula for calculating the offset is given below:

$$\text{offset} = [(row - 1) * n + (column - 1)] * \text{byte size of each element}$$

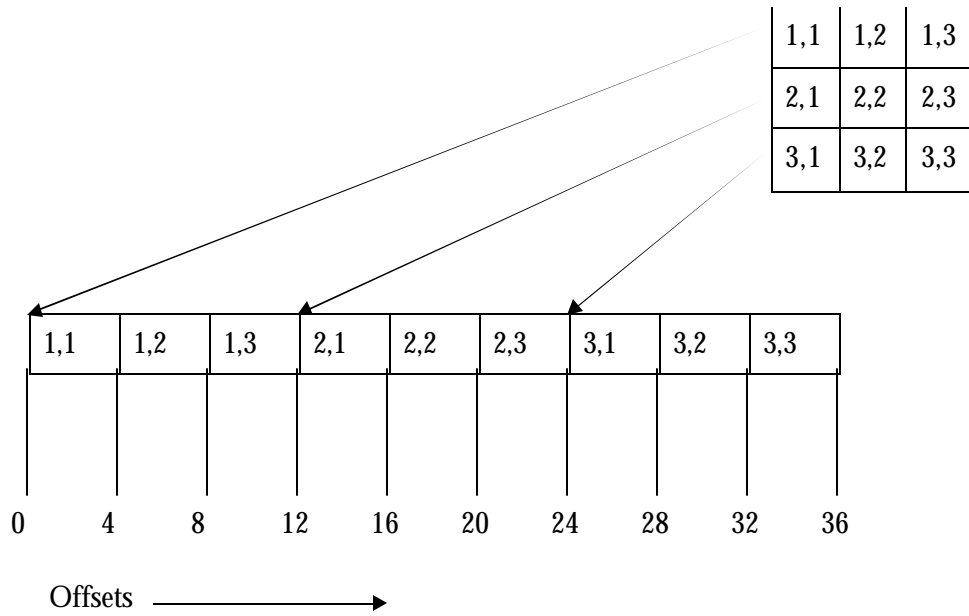


Figure 5.1 Organisation of GridFile

5.3.2 Wavelet Transform

The wavelet transform is a signal-processing technique that gives the frequency content of a signal by filtering. The filtering is done by convolving the input signal s with an appropriate filter.

Suppose that s is the input signal, f is the filter to be used and l is the length of the filter. If s is the output signal, then by convolving s with f we get:

$$s_i = \sum_{k=1}^l f_k s_{i+k-l/2}$$

Formula 5.1 Wavelet Transform

We can choose an appropriate filter from a number of options. The popular Cohen-Daubechies-Feauveau (2,2) biorthogonal wavelet is shown below:

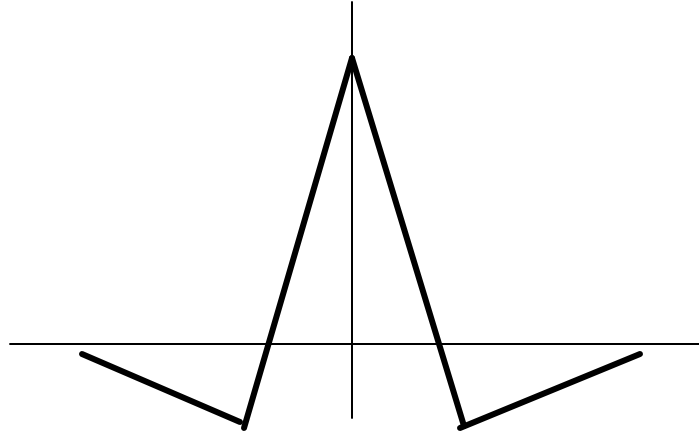


Figure 5.2 Cohen -Daubechies-Feauveau (2,2)

In our project, we preferred to use a hat-shaped filter such as the one shown below:

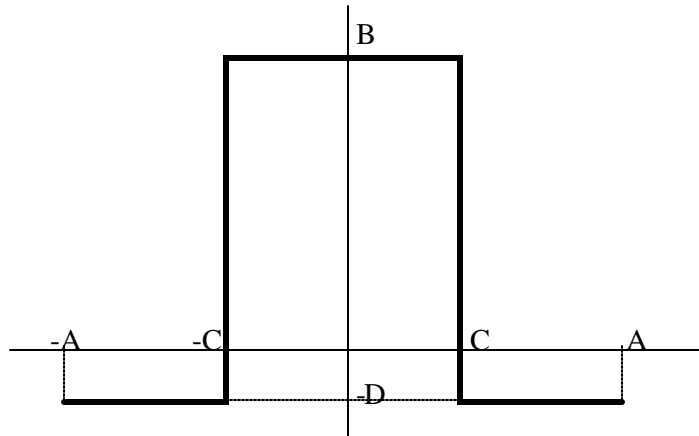


Figure 5.3 Hat-Shaped Filter

- Algorithm *WaveletTransform*

Let *QuantisedFile* be the output $m * n$ GridFile generated by the Quantisation stage. Let *Coefficient* be the array of filter coefficients. Let N be the number of filter coefficients. Let A , B , C and D be the filter parameters as shown in figure 5.3. Let *Threshold* denote the threshold value used to perform the filtering. Let *FilteredFile* be the $m * n$ GridFile output by the *WaveletTransform* procedure.

1. $xinc = 2 * A / (N - 1)$, $x = -A$
2. for $i = 1$ to N
 - if $(A \leq x < -C)$ *Coefficient* [i] = $-D$
 - if $(C \leq x \leq C)$ *Coefficient* [i] = B
 - if $(C < x \leq A)$ *Coefficient* [i] = $-D$
 - $x += xinc$
3. for $i = 1$ to m
 - for $j = 1$ to n
 - $temp = 0$
 - for $k = 1$ to N
 - $temp = temp + Coefficient[k] * QuantisedFile[i][j - N / 2 + k]$
 - $temp = temp + Coefficient[k] * QuantisedFile[i - N / 2 + k][j]$
 - $temp = temp + Coefficient[k] * QuantisedFile[i - N / 2 + k][j - N / 2 + k]$
 - $temp = temp + Coefficient[k] * QuantisedFile[i + N / 2 - k][j - N / 2 + k]$
 - if $(temp \geq Threshold)$
 - $FilteredFile[i][j] = 1$
 - else
 - $FilteredFile[i][j] = 0$

5.3.3 Connected-component Detection

Detection of connected components between pixels in the binary image generated by the clustering algorithm is a fundamental step in the segmentation of the image into clusters. Each cluster is assigned a unique label to separate it from other clusters. All the pixels within a cluster of spatially connected 1's are assigned the same label.

The basic algorithm performs two passes through the image. In the first pass, the image is processed from left to right and top to bottom to generate labels for each pixel and all of the equivalent labels are stored in a pair of arrays. In the second pass, each label is replaced by the label assigned to its equivalence class. However, for large images, the equivalence arrays can become unacceptably large. The algorithm implemented in our project uses the divide-and-conquer approach to finding connected regions.

5.3.3.1 Basic Pixel Connectivity

A pixel p at coordinate (x, y) has four direct neighbours, $N_4(p)$ and four diagonal neighbours, $N_D(p)$. Eight-neighbours, $N_8(p)$ of a pixel p consist of the union of $N_4(p)$ and $N_D(p)$. To establish connectivity between pixels of 1's in a binary image, three type of connectivity for pixels p and q can be considered:

1. 4-Connectivity:
Connected if q is in $N_4(p)$
2. 8-Connectivity:
Connected if q is in $N_8(p)$
3. m-Connectivity:
Connected if q is in $N_4(p)$, or if q is in $N_D(p)$ and $N_4(p) \cap N_4(q) \neq \emptyset$.

The labeling algorithm used in our project is based on 8-connectivity.

5.3.3.2 Connected Component Labelling Algorithm

- *Step 1: Initial Labelling*

Scan the image pixel by pixel from left to right and top to bottom. Let p denote the current pixel in the scanning process and $4-Nbr$ denote the four neighbouring pixels in the W, NW, N and NE direction of p .

- If p is 0, move to the next scanning position.
- If p is 1 and all values in $4-Nbr$ are 0, assign a new label to p .
- If only one value in $4-Nbr$ is not 0, assign its value to p .
- If two or more values in $4-Nbr$ are not 0, assign one of the labels to p and mark labels in $4-Nbr$ as equivalent.

- *Step 2: Resolve Equivalences*

The equivalent relations are expressed as a binary matrix L . Equivalence relations satisfy reflexivity, symmetry and transitivity. To add reflexivity in matrix L , all main diagonal elements are set to 1. To obtain transitive closure, the Floyd-Warshall algorithm is used.

```

for  $j = 1$  to  $n$ 
  for  $i = 1$  to  $n$ 
    if  $L[i, j] = 1$  then
      for  $k = 1$  to  $n$ 
         $L[i, k] = L[i, k] \text{ OR } L[j, k]$ 

```

This algorithm can be performed using $O(n^3)$ **OR** operations. After calculating the transitive closure, each label value is recalculated to resolve equivalences. The image is scanned again and each label is replaced by the label assigned to its equivalence class.

	1	2	3	4	5	6
1		1				1
2	1					
3				1		
4			1		1	
5				1		
6	1					

Before

	1	2	3	4	5	6
1	1	1				1
2	1	1				1
3			1	1	1	
4			1	1	1	
5			1	1	1	
6	1	1				1

After

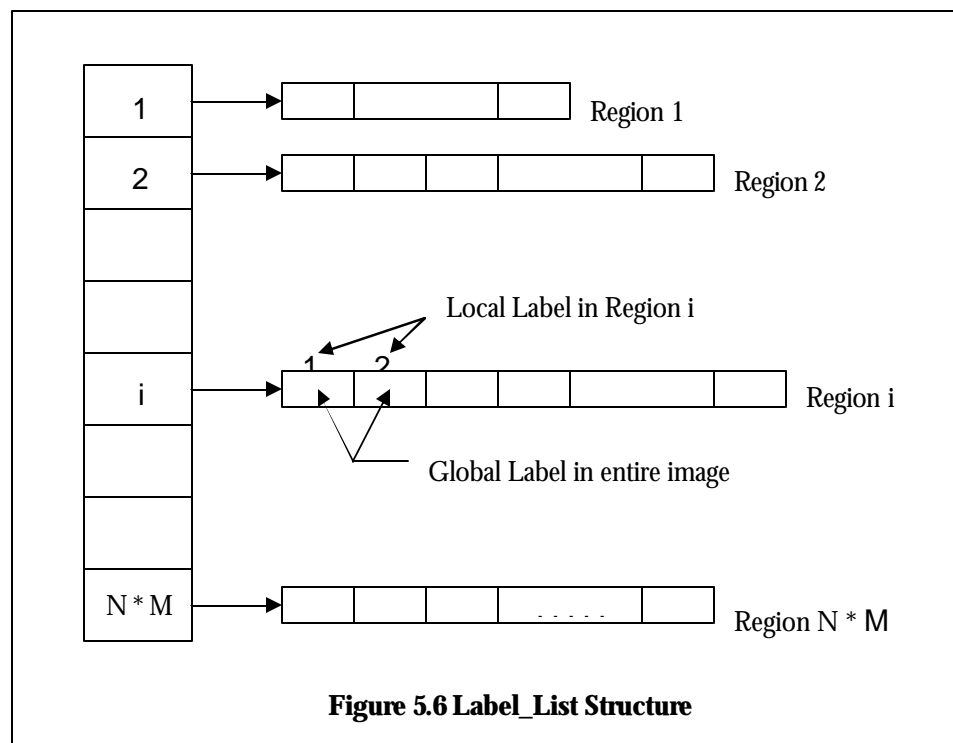
Figure 5.4 Resolving Equivalences

5.3.3.3 Divide-and-Conquer Approach

We divide the image into $N * M$ small regions. The large equivalence array is the main bottleneck in the original algorithm, but $N * M$ small equivalence arrays can be found in greatly reduced time. We then connect each region with its neighbour regions to generate the actual label within the entire image. We use $N * M$ pointers $Label_List[i]$ to point to arrays that maintain the actual label within the entire image and the index of the region i .

Region 1	Region 2	Region 3	Region 4
Region 5	Region 6	Region 7	Region 8
Region 9	Region 10	Region 11	Region 12

Figure 5.5 Image divided into $4 * 3$ regions



- *Step 1:*
Divide the given image into $N * M$ small regions and set $Total_Index = 0$
- *Step 2:*
For each region $i = 1$ to $N * M$
 - Apply *Step 1* of the basic algorithm to the selected region
 - Allocate memory for the array pointed to by $Label_List[i]$ as maximum number of labels for Region i
 - Apply *Step 2* of basic algorithm to resolve the equivalences for Region i
 - For $j = 1$ to Maximum number of labels for Region i
 $Label_List[i][j] = Total_Index + Label$
 Where $Label$ is the label assigned to the corresponding equivalence class after equivalence resolution
 - $Total_Index = Total_Index + \text{maximum}\{Label\}$
 - If $i > 1$ then call $Merge(i)$
- *Step 3:*
For each region $i = 1$ to $N * M$, scan the image in Region i from left to right, top to bottom and replace the local label with value k with $Label_List[i][k]$

The Merge function is used to resolve equivalences between adjoining regions.

- *Merge(i) Function*
 - *Step 1:*
Select first pixel p in Region i
 If $label(p) > 0$ then
 For each pixel q in $N_s(p)$ that is in another region
 If $label(q) > 0$ then
 Call $Resolve_Equivalence(p, q, i)$

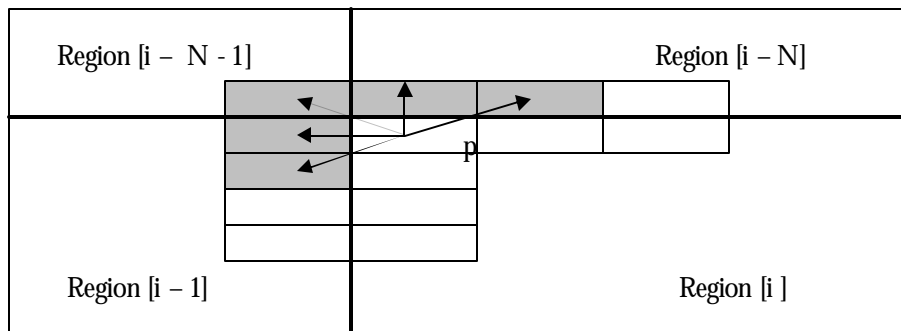


Figure 5.7 Merging Left-Top Corner Pixel

- *Step 2:*
 For each pixel p in the first column of Region i
 If $label(p) > 0$ then
 For each pixel q in $N_s(p)$ that is in Region $i - 1$
 If $label(q) > 0$ then
 Call *Resolve_Equivalence* (p, q, λ)

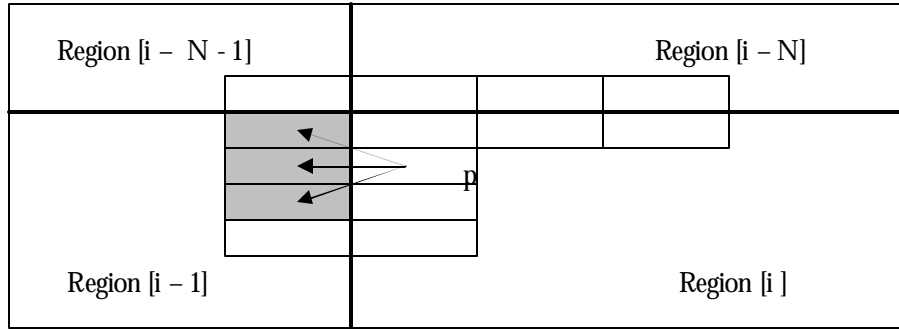


Figure 5.8 Merging Leftmost Column Pixels

- *Step 3:*
 For each pixel p in the first column of Region i
 If $label(p) > 0$ then
 For each pixel q in $N_s(p)$ that is in Region $i - N$
 If $label(q) > 0$ then
 Call *Resolve_Equivalence* (p, q, λ)

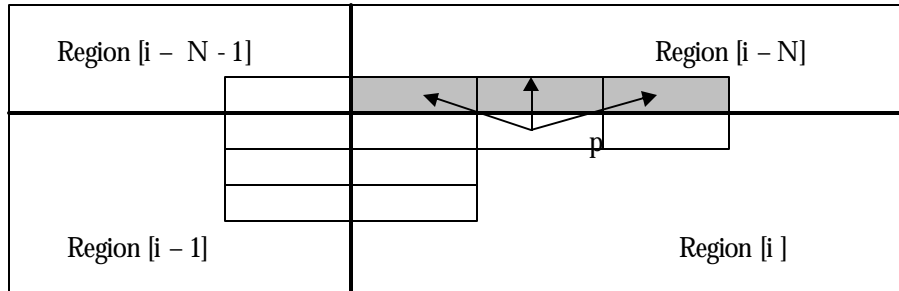


Figure 5.9 Merging Topmost Row Pixels

The *Resolve_Equivalence* Function is used to resolve two equivalent labels p and q .

- *Resolve_Equivalence* (p, q, λ) Function
 - *Step 1:*
 - $Index1 = Label_List [Region\ Number\ of\ q][label(q)]$
 - $Index2 = Label_List [i][label(p)]$
 - If ($Index1 \neq Index2$)
 Perform *Step 2*

- *Step 2:*
 - $Small_Label = \text{minimum} \{ Index1, Index2 \}$
 - $Large_Label = \text{maximum} \{ Index1, Index2 \}$
 - For $k = 1$ to i
 - For $j = 1$ to Size of Array for Region k
 - If $(Label_List[k][j] > Large_Label)$ then
 - $Label_List[k][j] = Label_List[k][j] - 1$
 - Else if $(Label_List[k][j] = Large_Label)$ then
 - $Label_List[k][j] = Small_Label$
 - $Total_Index = Total_Index - 1$

5.3.4 Extraction of Cluster Components

The output of the Connected-components detection algorithm is a GridFile which consists of densely packed groups of numbers. Each group represents a cluster of stars and each group has a unique identifier so that we can differentiate between individual clusters. As the file represents the area of interest, the file gives us the geometric locations of each cluster. Hence, we can map each cluster from the file to an area in the sky. The stars that lie within this area are said to form the cluster, i.e. these stars are the components of that cluster.

The *SearchedFile* gives us a list of all stars that lie within the area of interest. Hence, each star either is a part of a cluster, or it is an outlier that is to be discarded. The GridFile divides the area of interest into identical cells and each cell will have a label which is non-zero if the area is part of a spatial cluster. To determine the components of every cluster, we take each star from *SearchedFile* and determine the cell within which this star must lie. If the label of that cell is zero, the star can be ignored. If it is non-zero, say n , then the star is part of cluster n .

Once this process is completed, we have a list of components for every cluster and these lists can be saved to a file for later use. The user can select the clusters that he wishes to save on the file. The lists corresponding to those clusters are combined and written to a single output file for later retrieval and analysis.

- Algorithm *ClusterExtraction*

Let *ConnectedFile* be the output $m * n$ GridFile generated by the Connected Component Labelling algorithm. Let N be the total number of clusters which is initialised to *Total_Index*. We use an array of temporary files *TempFile* of length N to store the components of each cluster separately. Let *Count* be an array of length N which contains the number of stars in each cluster. Let *IsolatedFile* be the output file generated by this algorithm.

1. For every entry in *SearchedFile*
 - Retrieve a record number from *SearchedFile*.
 - Read the corresponding star record from the main database.
 - Let RA and DEC be the values of Right Ascension and Declination of the corresponding star.
 - $GridX = (RA - RA_{min}) / GridSizeX$
 - $GridY = (DEC - DEC_{min}) / GridSizeY$
 - If *ConnectedFile* [$GridX$][$GridY$] = i , $i \neq 0$
 - Store the record number of the corresponding star in *TempFile* [i]
 - $Count[i] = Count[i] + 1$
2. Store the search area and the grid sizes in *IsolatedFile*.
3. Store the number of clusters N in *IsolatedFile*.
4. for $i = 1$ to N
 - Store the number of stars in cluster i , that is $Count[i]$ in *IsolatedFile*.
5. for $i = 1$ to N
 - Store the record numbers of all stars in cluster i from *TempFile* [i] in *IsolatedFile*

5.4 Overview of WaveCluster Algorithm

Now that we have seen the details of the WaveCluster algorithm, we can give an overview of the entire process with the help of the following illustrations.

1. Consider the area to be mined specified by a particular search window shown in figure 5.9.

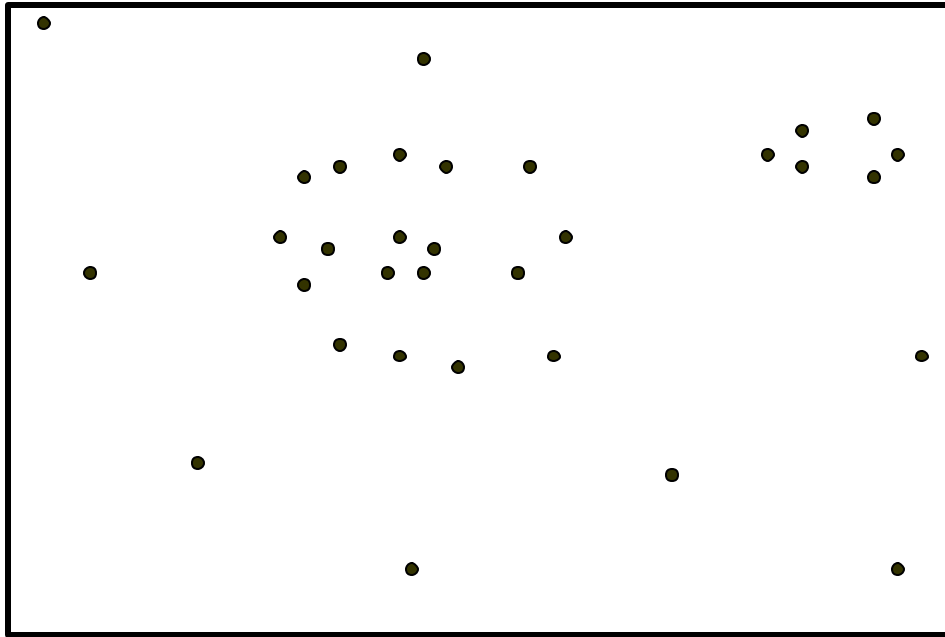


Figure 5.10 Search Area

2. We divide the area of interest into identical cells.

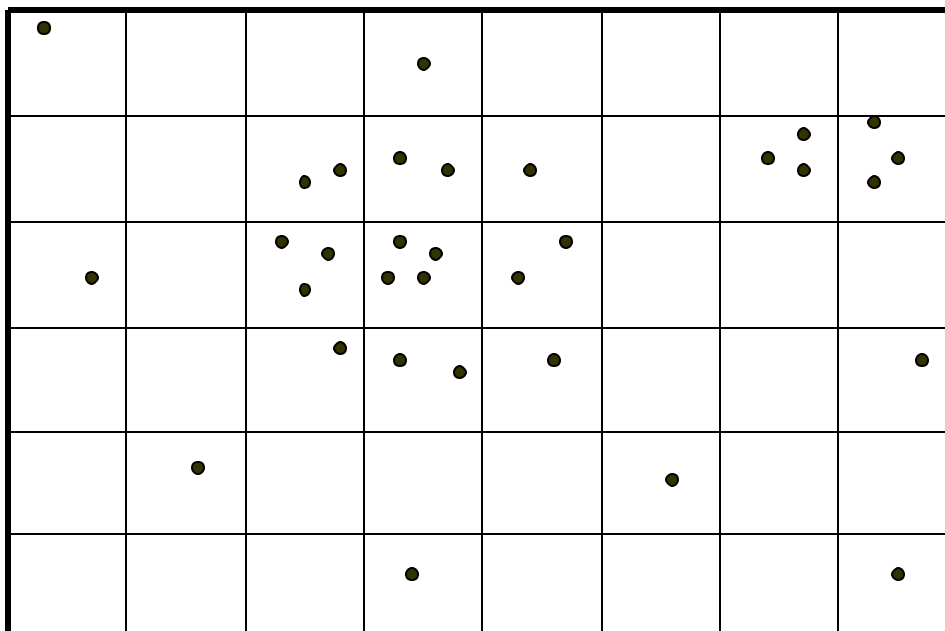


Figure 5.11 Dividing into Cells

3. This is followed by the quantisation of the area.

1	0	0	1	0	0	0	0
0	0	2	2	1	0	3	3
1	0	3	4	2	0	0	0
0	0	1	2	1	0	0	1
0	1	0	0	0	1	0	0
0	0	0	1	0	0	0	1

Figure 5.12 Quantisation of the Area

4. The next step is transformation of the area.

0	0	0	1	0	0	0	0
0	0	1	1	1	0	1	1
0	0	1	1	1	0	0	0
0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figure 5.13 Transformation of the Area

5. Further the connected regions algorithm is applied to get the individual clusters.

0	0	0	1	0	0	0	0
0	0	1	1	1	0	2	2
0	0	1	1	1	0	0	0
0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figure 5.14 Applying Connected Regions Algorithm

CHAPTER SIX

Cluster Analysis

In order to confirm that the detected spatial clusters are valid, it is necessary to analyse the detected clusters for their properties. Once it has been established that the program can detect the existing clusters, we can further extend the application towards finding of new clusters. The various techniques for analysis include the following.

6.1 The Hertzsprung-Russell (H-R) Diagram

The Hertzsprung-Russell (H-R) Diagram, pioneered independently by Einar Hertzsprung and Henry Norris Russell, plots Luminosity as a function of Temperature for stars. In general, it is a graph that is based on the structure shown in figure 6.1 below.

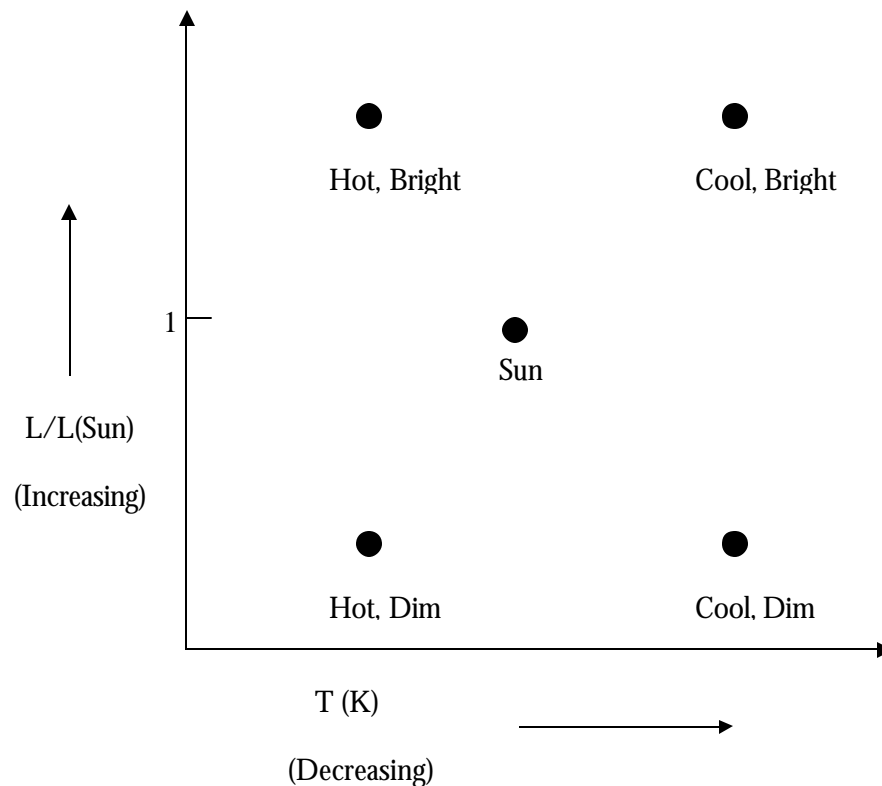


Figure 6.1 Basic Structure of H-R Diagram

6.1.1 Structure of H-R Diagram

In an H-R diagram, each star is represented by a dot. The position of each dot on the diagram corresponds to the star's luminosity and its temperature. The vertical position represents the star's luminosity which could be the luminosity in Watts, but more commonly it is in units of the Sun's luminosity. The horizontal position represents the star's surface temperature usually labelled by the temperature in Kelvin's. It is traditional to have the highest Temperatures go to the left. Instead of temperature sometimes the star's spectral class (OBAFGKM)⁹ or B-V colour index can also be used along the horizontal axis.

It is readily apparent that the H-R Diagram is not uniformly populated, but that stars preferentially fall into certain regions of the diagram. The majority of stars fall along a curving diagonal line called the *Main Sequence* (the point where the stars begin burning hydrogen in their centers), but there are other regions where many stars also fall.

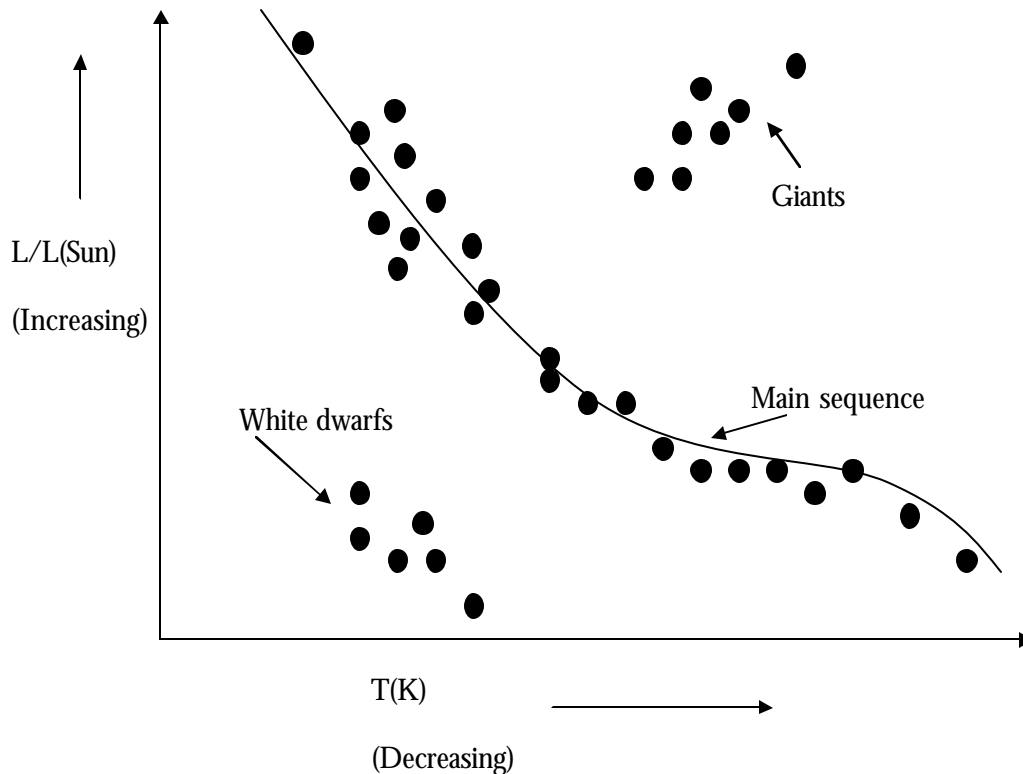


Figure 6.2 H-R Diagram Pattern

⁹ Refer appendix for details on spectral classes.

The most conspicuous region of the H-R diagram is the sequence of stars running from extremely bright, hot stars in the upper left-hand corner to faint, cool stars in the lower right-hand corner. This sequence is called the *main sequence*, and it contains most of the stars that could be plotted on the diagram.

The second most prominent region in the H-R diagram is the region labeled as *red giants*. They are luminous stars lying above the main sequence in a region that angles up toward the upper right-hand corner. On average, they are 100 times more luminous than the Sun, and they vary in surface temperature from 3000 to 7000 K.

Stars are called *red supergiants* if they lie on the cool side of the diagram and *blue supergiants* if they are early-type stars of classes O and B. The red supergiants and the blue supergiants can be hundreds of thousands of times more luminous than our Sun.

The last region of importance contains faint stars lying below the main sequence; these are called *white dwarfs*. White dwarfs are typically a few thousandths of the luminosity of the Sun, even though their outer layers are hotter than those of the Sun.

6.1.2 Uses of H-R Diagram

When stars forming any cluster are plotted to obtain HR diagram pertaining to that cluster, it reveals number of characteristics of the whole cluster and also physical, chemical properties of the constituent stars.

- HR diagram can be used to determine *age of the open cluster*. Stars in a cluster are formed at the same time, in the same molecular cloud. Therefore, stars in a cluster
 - have the same age
 - had the same initial chemical composition,
 - are at roughly the same distance from Earth.

Thus, when stars form within a cluster, they differ only in their mass. The more massive stars evolve more rapidly, so to find the age of a cluster of stars, we need to determine the mass of the stars which have just now exhausted the hydrogen in their cores and are turning into red giants.

- H-R diagram can be used to determine the rough distance to the cluster. By identifying the dwarfs from the main sequence and using parallax measure, a rough estimate about the clusters distance can be made.

- The H-R Diagram will also help in differentiating between *Open Clusters* and *Globular Clusters*. Different types of clusters display different main sequence patterns. For example the Pleiades which is an open cluster displays a characteristic main sequence as shown in the figure 6.3 below:

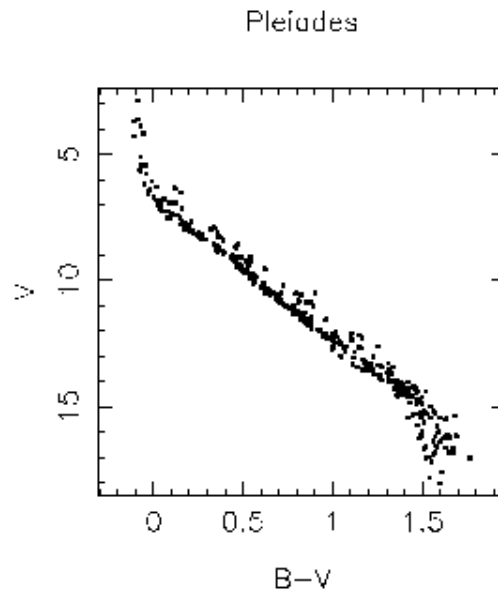


Figure 6.3 H-R Plot for Pleiades

Thus by identifying the main sequence pattern an estimate about the characteristic of the cluster can be made. The HR plot generated by our program is shown below. By comparing, we see that it agrees fairly with the actual H-R plot.

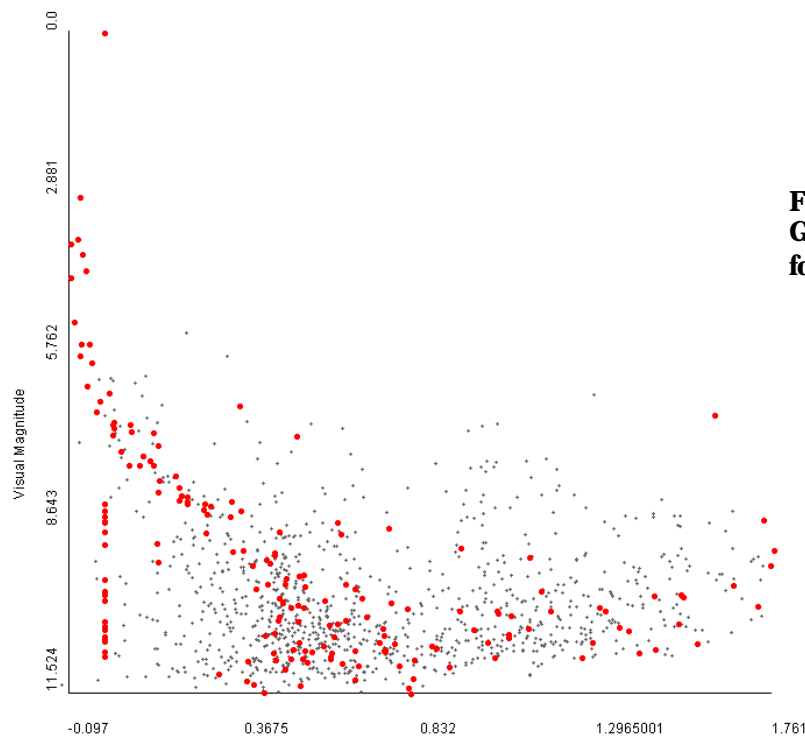


Figure 6.4 Program Generated H-R Plot for Pleiades

6.2 Cluster Properties

The stars that constitute the detected spatial cluster are stored in a file. These stars can be compared with the existing catalogues that store information about the stars constituting an existing cluster. Some of the important properties that will help us in this analysis and verification process are as follows –

- Number of stars in the cluster
- Mean RA and mean DEC of the stars in the cluster.
- Average luminosity or temperature of the cluster
- Mean Parallax of the stars in the cluster.
- Span of the cluster.

CHAPTER SEVEN

Visualisation

After mining the search window for clusters, the user has various options of getting a visual representation of the cluster on the *Visualisation Screen*¹⁰. This assists the user in determining the approximate shape and relative position of the cluster. The clusters are displayed using either the *Equatorial Co-ordinate System* or the *Galactic Co-ordinate System*. Options for Zooming and Panning of the user-defined window are also a part of the visualisation

7.1 Equatorial Co-ordinate System

The Equatorial System can be envisioned as the extension of our terrestrial latitude and longitude projected onto the celestial sphere. The coordinate analogous to latitude is called Declination (DEC), and the coordinate analogous to longitude is called Right Ascension (RA).

Using the RA and DEC, along the x and y axes respectively, we display all the stars in the user specified search window. Initially the search window will cover the entire visualisation screen. The user has the option of increasing and decreasing the window size relative to the display screen, hence providing the zooming options. Similarly, this window can be panned in the four directions using either the shortcuts or options from the menu. Further user has the options of displaying stars that do not form part of the cluster in a different colour. In case of multiple clusters within a search window, the user has the option of viewing each cluster individually. The various output screens for both Pleiades and Perseus clusters are shown in the following figures.

¹⁰ Refer to Section 8.2.1 for Visualisation Screen

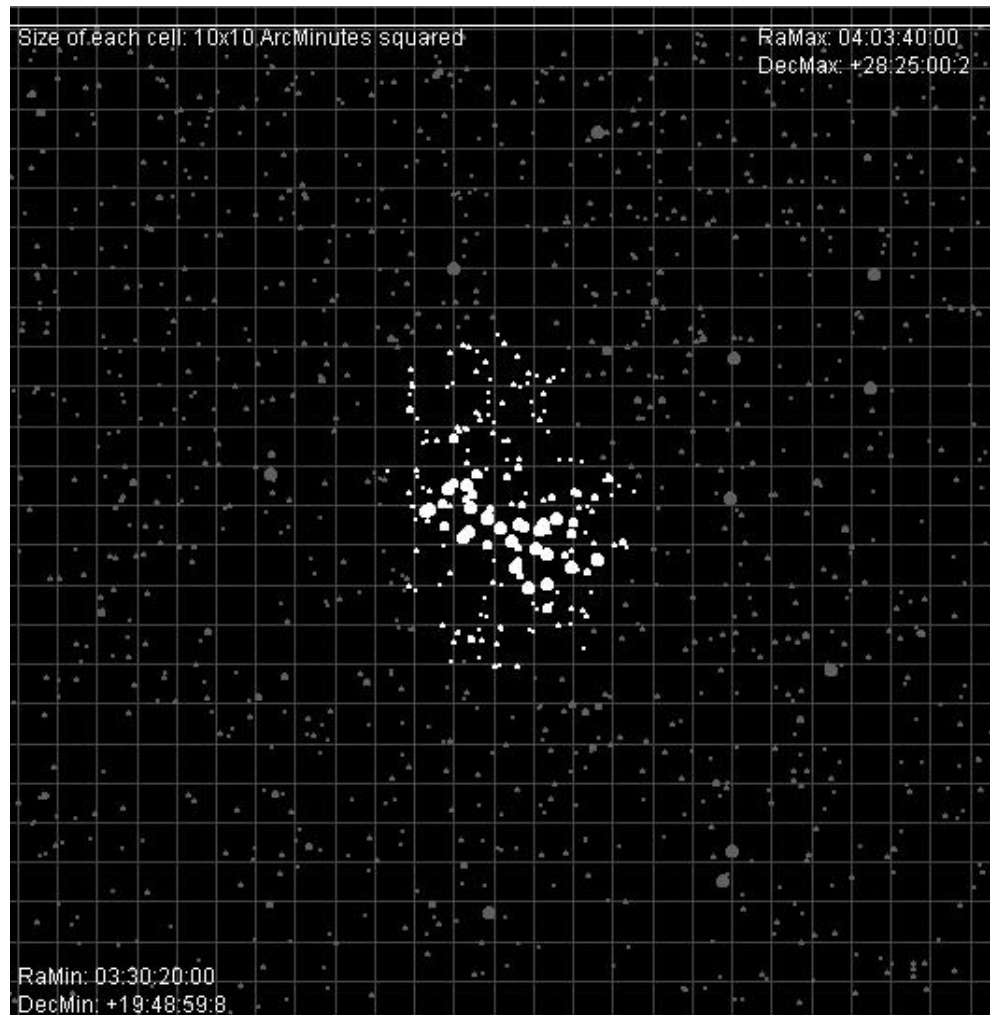


Figure 7.1 Pleiades Cluster With Background Stars

RAmin:	03:30:20.00
DECmin:	+19:48:59.8
RAmax:	04:04:40.00
DECmax:	+ 28:25:00.2

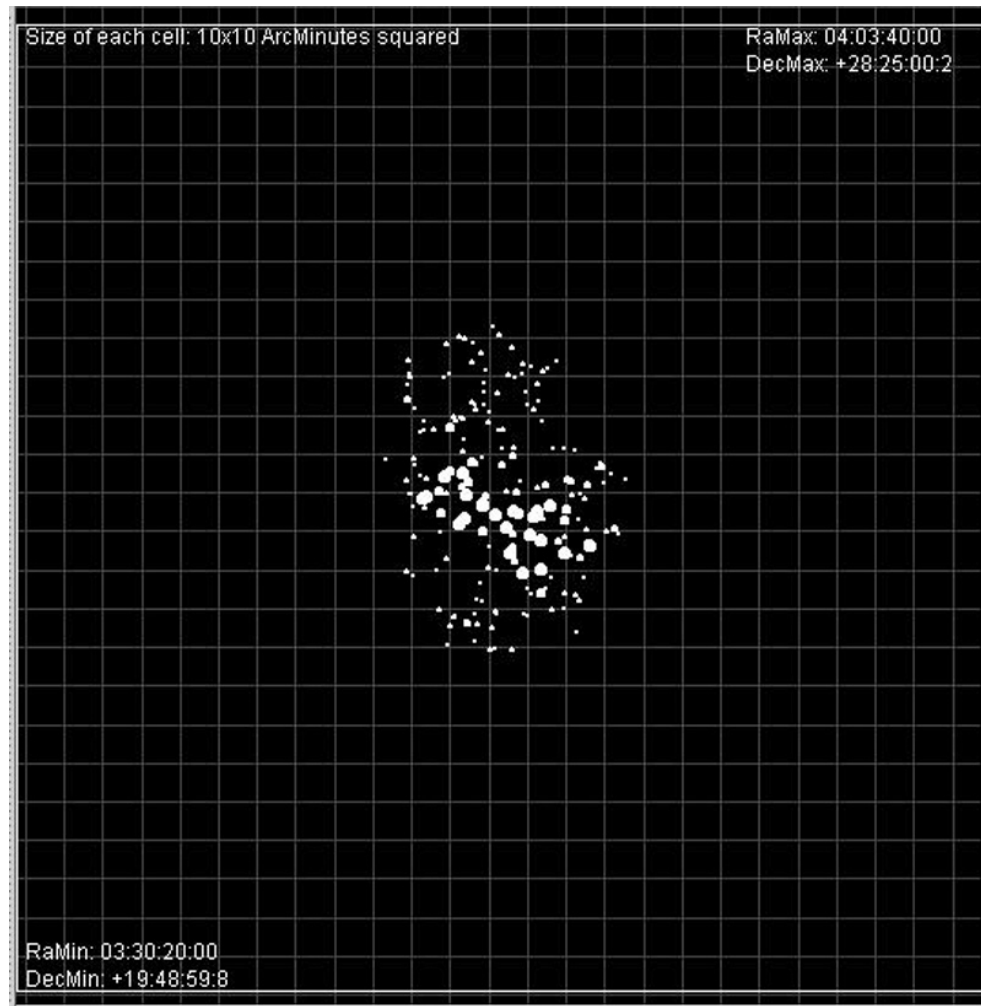


Figure 7.2 Pleiades Cluster Without Background Stars

RAmin:	03:30:20:00
DECmin:	+19:48:59.8
RAmax:	04:04:40:00
DECmax:	+28:25:00.2

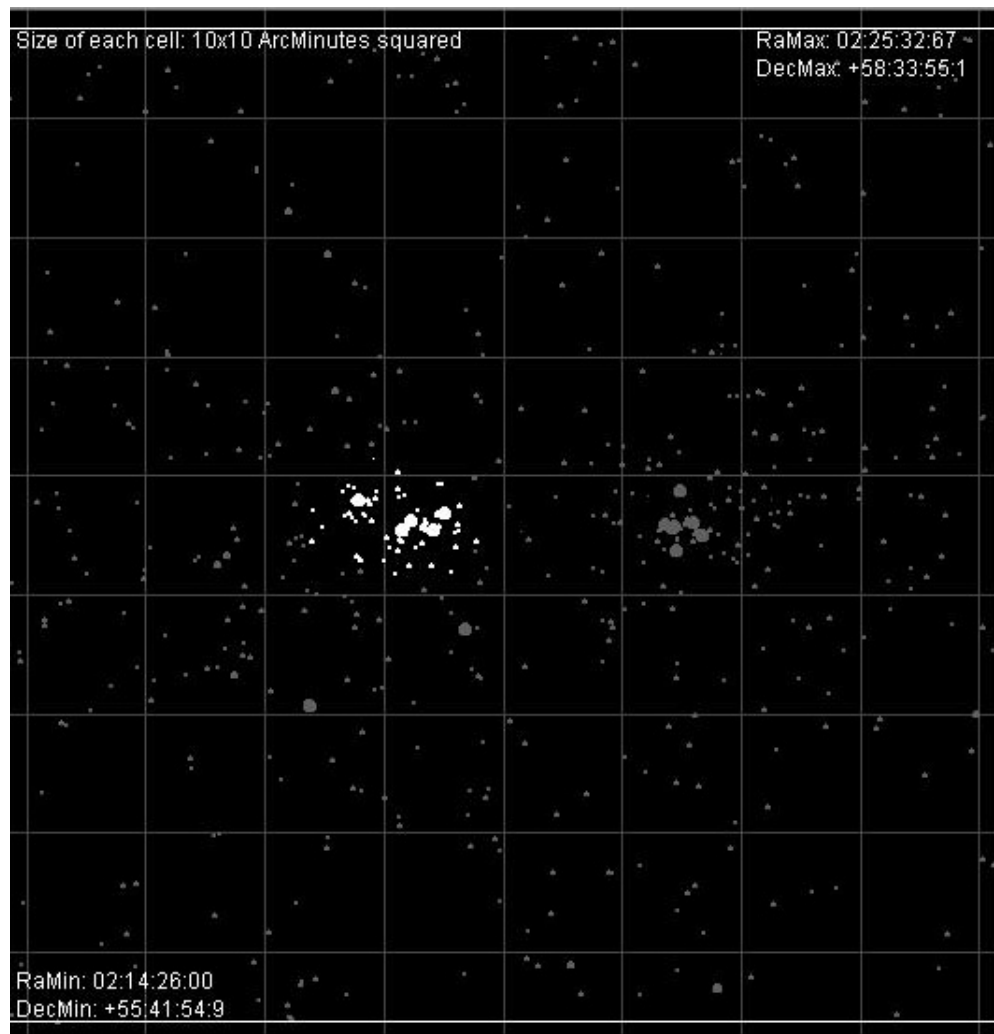


Figure 7.3 Perseus Double Cluster – Cluster No.1 Only

RAmin: 02:14:26:00
DECmin: +55:41:54.9
RAmax: 02:25:32:67
DECmax: +58:33:55.1

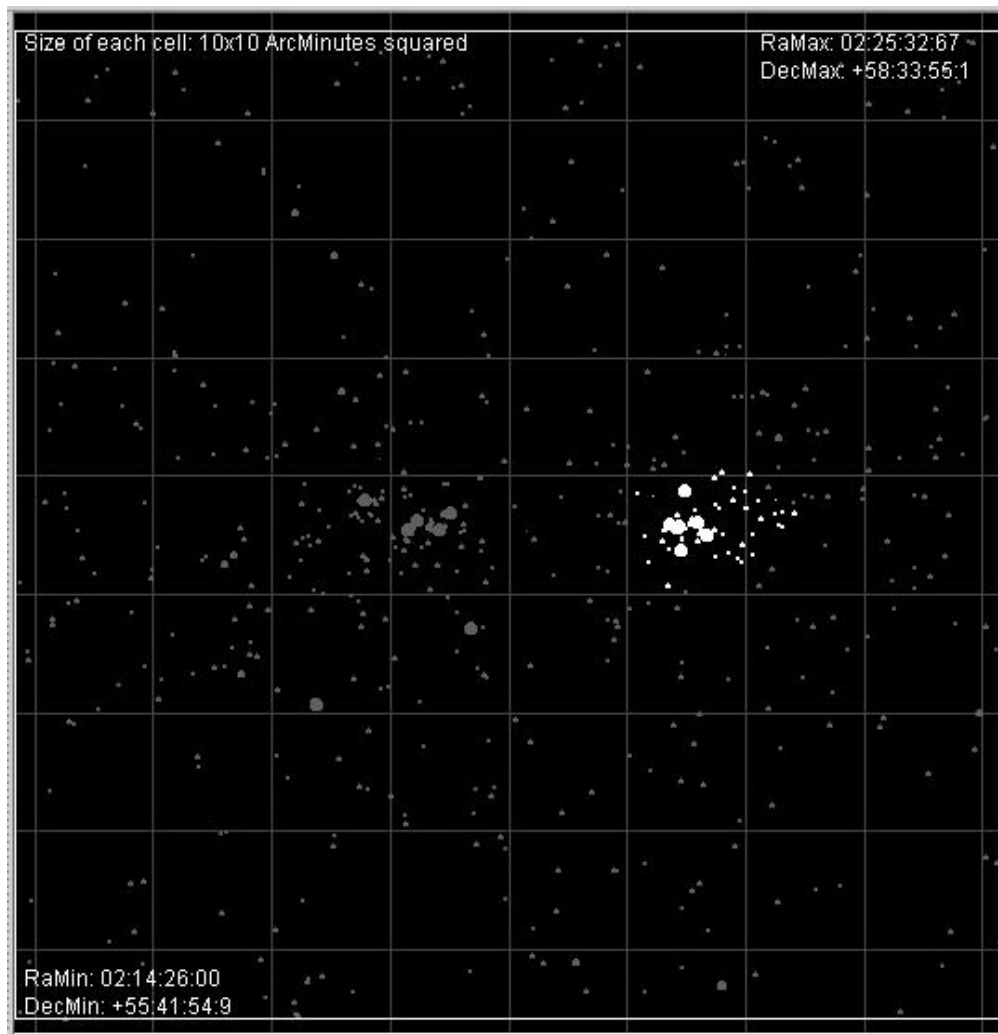


Figure 7.4 Perseus Double Cluster – Cluster No. 2 Only

RAmin: 02:14:26:00
DECmin: +55:41:54.9
RAmax: 02:25:32:67
DECmax: +58:33:55.1

7.2 Galactic Co-ordinate System

When studying objects in the sky it is helpful to use a coordinate system to keep track of where things are. The two specific coordinates are *galactic longitude* (l) and *latitude* (b). The disk of the galactic plane lies at $b = 0$ all around the sky. The galactic center defines $l=0$. Latitude is measured as the angle from the galactic plane (therefore, straight out of the plane is $b = 0$). Longitude is defined as angle from the Galactic center along a circle of constant latitude.

The user gets to view the cluster detected, in galactic system, by projecting it on a modified Pseudocylindrical Azimuthal, equal area plane, which is widely known as *hammer*. Figure 7.5 shows the galactic co-ordinate frame.

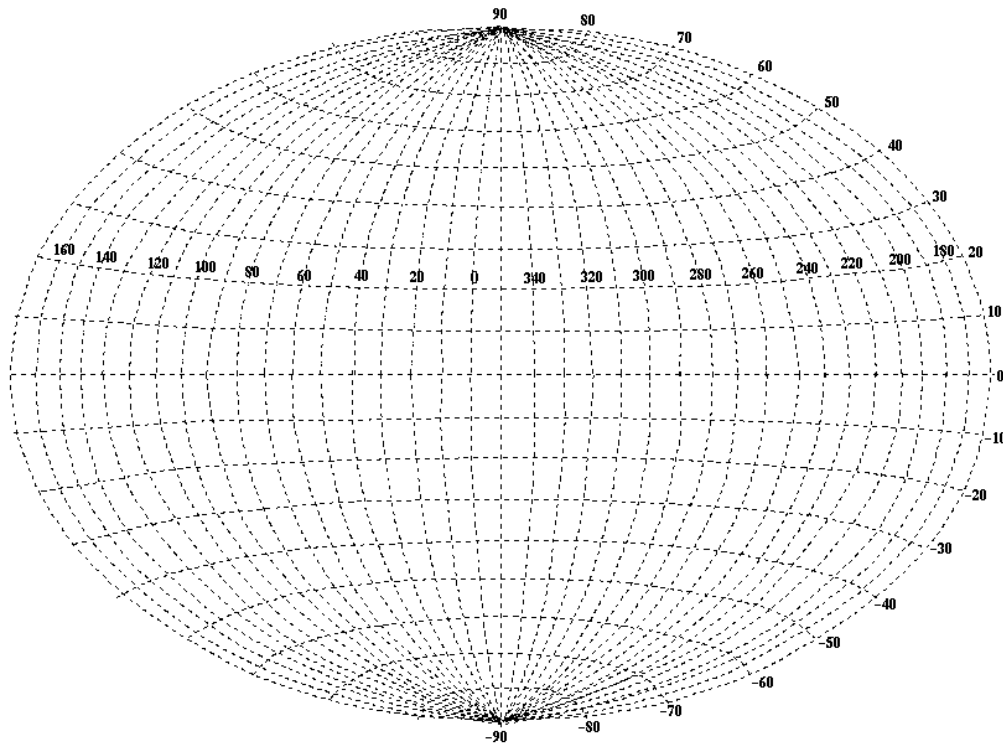


Figure 7.5 Pseudo Cylindrical Hammer

Since we are using Right Ascension and Declination of the stars for searching and clustering, we can obtain the corresponding galactic co-ordinates as shown.

Given Equatorial co-ordinates d (Declination) and a (Right Ascension), the galactic co-ordinates (b, l) can be computed using the following formulae –

- $\cos(b) \cdot \cos(l - 33^\circ) = \cos(d) \cdot \cos(a - 282.25^\circ)$
- $\cos(b) \cdot \sin(l - 33^\circ) = \sin(d) \cdot \sin(62.6^\circ) + \cos(d) \cdot \sin(a - 282.25^\circ) \cdot \cos(62.6^\circ)$
- $\sin(b) = \sin(d) \cdot \cos(62.6^\circ) - \cos(d) \cdot \sin(a - 282.25^\circ) \cdot \sin(62.6^\circ)$

After conversion to the galactic co-ordinates, the next step is the projection on the pseudocylindrical hammer frame. This is done by using the following formulae for getting the x and y co-ordinates.

Given the galactic co-ordinates $(glat, glong)$ and radius R , we get the plotting co-ordinates as (x, y) :

- $x = \frac{R \cdot 2 \cdot \sqrt{2} \cdot \cos(glat) \cdot \sin(glong / 2)}{\sqrt{1 + \cos(glat) \cdot \cos(glong / 2)}}$
- $y = \frac{R \cdot \sqrt{2} \cdot \sin(glat)}{\sqrt{1 + \cos(glat) \cdot \cos(glong / 2)}}$

The output screen for the relative position and orientation for the Pleiades cluster is shown in figure 7.6. The stars in red are considered to be part of the cluster whereas the stars in white are forming the background and within the search window.

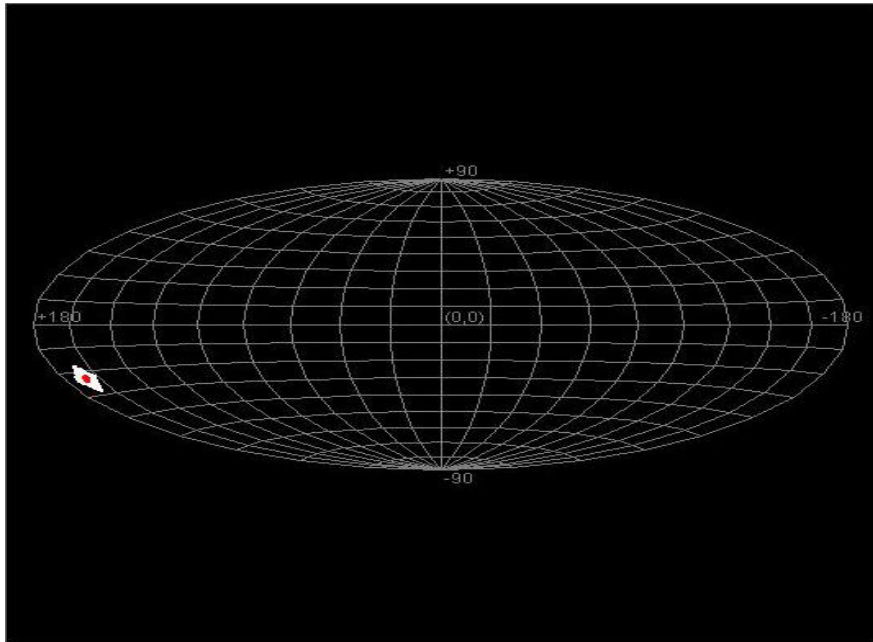


Figure 7.6 Pleiades Cluster- Galactic Plot

CHAPTER EIGHT

User Interface

The Graphical User Interface (GUI) is the face of the project that is presented to the user. Hence, the design of the GUI is an important aspect. We have kept the following principles in mind to develop a simple, neat and friendly user interface:

- All the functionality of the application must be provided by the interface.
- The user must be able to anticipate the behaviour of the various components, from their visual properties.
- Navigation should be simple and straightforward.
- Application must be made self-evident by the use of comprehensive help manuals and tips.
- Interface must be designed, so that it is transparent.
- Keyboard support in the form of short cut keys is essential.
- Providing too many functions at the top level is not a good idea.
- Avoiding modal behaviours that force the user into performing tasks in a fixed sequence. Allow the user to control the interactive flow.
- Maintenance of consistency between information display and data input.
- Deactivation of commands that are inappropriate in the context of the current actions.
- Providing visual and audio feedback to the user.

8.1 GUI Design Model

The GUI for the project has been based on the lines of the Model-View-Controller (MVC) Architecture. It is an idealized way of modelling a component as three separate parts:-

- The model that stores the data, which defines the component.
- The view that creates the visual representation of the component from the data in the model.
- The controller that deals with user interaction with the component and modifies the model and/or the view in response to a user action as necessary.

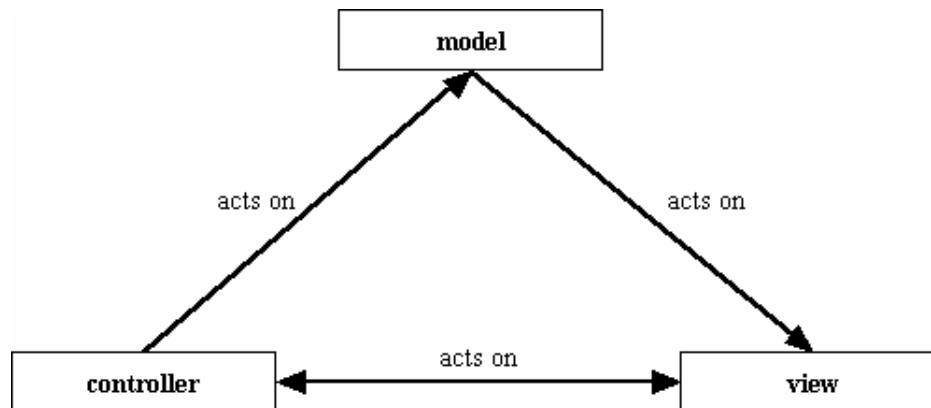


Figure 8.1 Model-View-Controller Architecture

8.2 Application Window

The application window as seen from the figure, consists of three parts i.e the menu, control panel on the left and the visualisation screen which covers the rest of the screen.

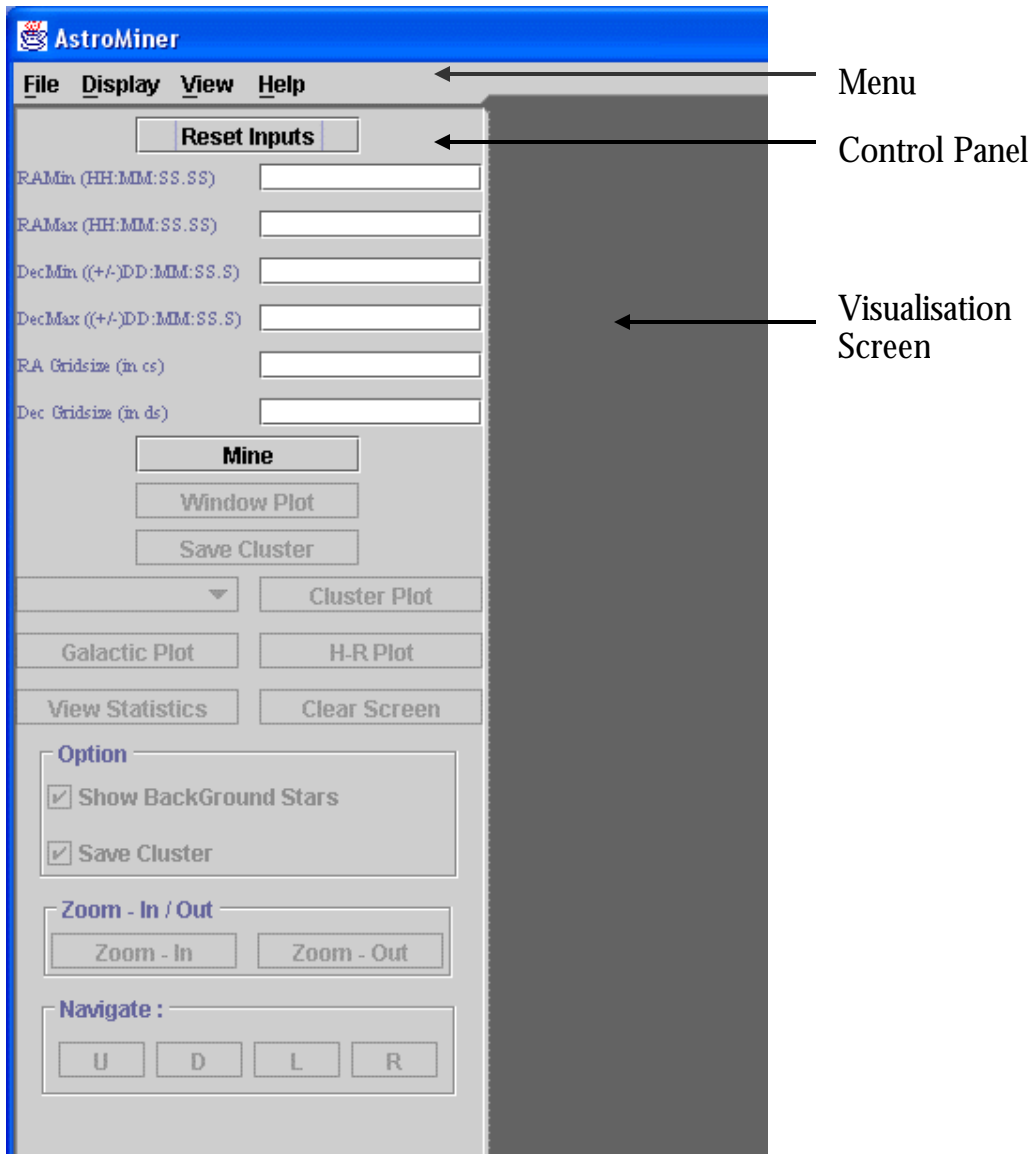


Figure 8.2 Graphical User Interface

8.2.1 Visualisation Screen

This is a very important part of the project interface and is used for outputting to the user the various plots as requested. Depending upon the option selected the visualisation screen will be dynamically updated to display the various plots.

8.2.2 Menu

The interactive menu and the control panel provide all the functionality, supported by the software. The menu options are listed below:

1. File

- **New:**
This option is used when the user wants to mine area of the sky by specifying a search window.
- **Load Mined Data:**
The user can load data of a previously saved cluster and directly use the visual tools without having to mine the sky for the search window again.
- **Browse Clustered Data:**
This allows the user to specify search windows of known astronomical star clusters.
- **Save As:**
This option allows the user to save the cluster data in a file with an extension of .clu.
- **Exit:**
This option allows the user to close the application.

2. Display

Once the mining has been done, or after loading of the cluster data, the visualisation tools are activated.

- **Window Plot:**
This option allows the user to plot the entire window with or without background stars.
- **Cluster Plot:**
This option allows plotting of clusters detected individually with or without the background stars.
- **Galactic Plot:**
This option allows the user to plot the position of cluster with respect to the galactic world frame.

- **HR Plot**

This option allows the user to plot the HR plot of the clusters detected.

- **View Statistics:**

This option allows the user to see some of the statistical properties for the cluster detected.

- **Clear Screen:**

This option allows the user to blank out the screen.

3. View

Manipulation of the plot, seen on the visualisation screen is done with the zooming and panning tools.

- **Zoom-In:**

This option allows the user to zoom in to the window specified.

- **Zoom-Out:**

This option allows the user to zoom out of the window specified.

- **Pan Left:**

This option allows the user to move the specified window left.

- **Pan Right:**

This option allows the user to move the specified window right.

- **Pan Up:**

This option allows the user to move the specified window up.

- **Pan Down:**

This option allows the user to move the specified window down.

4 Help

- **Contents:**

A comprehensive well-documented help file is provided.

- **Version:**

This option provides details of the application version

- **Credits:**

This option provides the user with information related to the developers of the project.

8.2.3 Control Panel

The control panel on the left consists of the following sub-panels: -

1. Input Panel

This part of the control panel allows the user to enter the search window in terms of the following parameters:

- **RA Min** (HH:MM:SS.SS)
- **RA Max** (HH:MM:SS.SS)
- **Dec Min** ((+/-)DD:MM:SS.S)
- **Dec Max** ((+/-)DD:MM:SS.S)

Further for the mining, grid cell sizes along the horizontal and vertical directions need to be specified for the quantisation.

2. Tool Panel

The tool panel consists of buttons dealing with the various visualisation plots as discussed in the menu options.

3. Options Panel

The options panel consists of two check boxes:

- **Background Stars:**
When checked the plotting is done with the background stars, otherwise only the stars that form a part of the cluster are used for plotting.
- **Save Cluster:**
When checked the current cluster will be marked for saving into a cluster data file.

4. Zoom Panel

This panel provides the same zooming options as discussed above.

5. Navigation Panel

This panel provides the panning options as discussed above.

CHAPTER NINE

Future Areas

Our implementation of the WaveCluster algorithm has the following salient features:

- Due to our spatial indexing system, we can mine arbitrarily large astronomical catalogues and visualise the results without any drop in performance.
- Due to the efficiency of the algorithm, the entire mining process can be completed in less than a few seconds. Hence, we can carry out the mining in real time.
- The main-memory requirements of our implementation are very minimal.

Our project could be furthered in scope in the following areas:

9.1 Using Larger Catalogues

Our implementation is able to successfully detect known clusters such as Perseus and Pleiades. However the true potential and utilisation of the program will be realised only when it is used to discover unknown clusters. As the Hipparcos and Tycho catalogues represent only a fraction of stars in the sky, the clusters that can be detected are limited. Moreover, these catalogues have been studied thoroughly and hence the chance of detecting previously undiscovered clusters is very remote.

To make the detection of previously undiscovered clusters, we need to mine larger catalogues such as the Guide Star Catalogue II, which contains about 430 million objects. The same indexing technique can be applied to this catalogue as well. The height of the tree will be $\lceil \log_{10} (430000000) - 1 \rceil$, i.e. 8. This gives us a minimum of 8 node comparisons and a maximum of 160. Hence we see that there is a negligible increase in the overhead while searching.

The time taken by the indexing procedure on the other hand will increase significantly. The Hipparcos catalogue took 1 hour to index, while the Tycho catalogue took 4.5 hours.¹¹ Though the indexing procedure on the Guide Star catalogue will take much longer, this is a one-time expenditure of effort. If we still wish to speed up the indexing procedure, we could make use of two alternatives:

¹¹ On a P-III 800 MHz machine

- We could use threads in Java to isolate the overhead of I/O operations from the computational operations
- If we had access to a parallel processing environment, we could modify our algorithm for index creation so that entries could be inserted in parallel, with different processors handling different sub-trees.

9.2 Mining the Image Data

The satellites, that provide information about the celestial bodies in the sky, provide this information in the form of image data, on the basis of which various catalogues are then constructed. The limitation of using astronomical catalogues for the mining process is that we are dependent on the accuracy and thoroughness of the catalogue. We can detect a new cluster only if the catalogue has the corresponding entries. However, most catalogues list stars only if they have a particular brightness. Hence even if a cluster may exist in a particular area of the sky, the catalogue may list only a few bright stars from within the cluster. Our algorithm will not be able to detect such clusters, since in fact they do not exist within the catalogue.

To overcome this, we could use the WaveCluster algorithm on the raw image data itself. As the image would already be available in the form of a bitmap, no conversion to feature space will be necessary, i.e. the Quantisation phase is made redundant. We could directly apply the Wavelet Transform and then detect the connected regions to identify the clusters.

The following points must be kept in mind while mining the raw image data:

- We need to identify the area of the sky that corresponds to the image being mined. As there is no way of identifying the area in terms of RA and DEC directly from the image, the user will have to provide these values.
- Clusters will be detected if they exist as densely packed regions in the feature space. While using catalogues, we could ensure this by adjusting the grid size. This problem is slightly different while using the image data. The image itself represents the feature space and if the inter-stellar separation in the image is very large, then there will not be any densely packed regions and hence, no clusters will be detected. Here, we need to make use of the multi-resolution property of the WaveCluster Algorithm, i.e. we need to adjust the resolution at which the mining is being carried out to enable accurate detection of clusters.

- Once the clusters are detected, we will be able to specify the geometric parameters of the cluster, such as location, size and shape. We will not be able to identify the individual components of the cluster. Knowing the location and size, we may then use a stellar catalogue to extract some entries that form part of the cluster.

9.3 Mining on Other Parameters

In our project, we have limited the mining process to the detection of spatial clusters. However, we could extend our project by carrying out the indexing on any two arbitrary parameters. We could then detect clusters on those parameters, however we would need a separate index for any two such pairs of parameters. Each index will take some time to create and each will occupy some memory. We could extend our interface to make the management of indices easy, by storing the index details and asking the user which index to use. The user then has the option of mining the data to detect spatial clusters, or clusters on any other parameters.

9.4 Mining in Multiple Dimensions

Our index structure as well as the feature space that we create is two-dimensional. Hence we can only detect clusters in two-dimensions, or in other words, we can only identify areas. However space and the actual clusters are three-dimensional. To account for this, we analyse our clusters using the H-R diagram to see if the cluster is a real cluster or not. However, this approach is limited as it depends upon the number of stars that lie within the cluster. If this number is small, then we cannot draw reliable inferences from the H-R diagram.

We could extend our indexing and mining techniques to multiple dimensions. If we wanted to carry out the mining in three dimensions, say RA, DEC and Parallax, we would have to index the catalogue on these fields and the mining process would also need to be extended to the third dimension. Once we did this, we would be able to identify volumes and hence the mining would be more reliable. Similarly, we could mine the catalogue on an arbitrary number of dimensions. However, the indexing time would increase exponentially, though the search time will not increase significantly. Also, the WaveCluster algorithm would take longer to run and there may be a significant delay while waiting for the results to be displayed.

LIST OF ILLUSTRATIONS

2.	Astronomical Data	
2.1	RA and DEC	5
2.2	Parallax	6
2.3	Proper Motion	6
3.	Indexed Access of Astronomical Data	
3.1	k-d Tree	9
3.2	QuadTree	9
3.3	Sample space and the corresponding R-Tree	11
3.4	Node Splitting	13
4.	Data Mining Techniques	
4.1	The Traditional KDD Paradigm	18
4.2	Organisation of Records in File	19
4.3	Processes in Data Mining	20
4.4	Clustering Algorithms	23
5.	The WaveCluster Algorithm	
5.1	Organisation of GridFile	28
5.2	Cohen-Daubechies-Feauveau (2,2)	29
5.3	Hat-Shaped Filter	29
5.4	Resolving Equivalences	32
5.5	Image divided into 4 * 3 regions	33
5.6	Label_List Structure	33
5.7	Merging Left-Top Corner Pixel	34
5.8	Merging Leftmost Column pixels	35
5.9	Merging Topmost Row Pixels	35
5.10	Search Area	38
5.11	Dividing into Cells	38
5.12	Quantisation of the Area	39
5.13	Transformation of the Area	39
5.14	Applying Connected Regions Algorithm	40

6.	Cluster Analysis	
6.1	Basic Structure of H-R Diagram	41
6.2	H-R Diagram Pattern	42
6.3	H-R Plot for Pleiades	44
6.4	Program Generated H-R Plot for Pleiades	44
7.	Visualisation	
7.1	Pleiades Cluster With Background Stars	47
7.2	Pleiades Cluster Without Background Stars	48
7.3	Perseus Double Cluster – Cluster No. 1 Only	49
7.4	Perseus Double Cluster – Cluster No. 2 Only	50
7.5	Pseudo Cylindrical Hammer	51
7.6	Pleiades Cluster- Galactic Plot	52
8.	User Interface	
8.1	Model-View-Controller Architecture	54
8.2	Graphical User Interface	55

LIST OF FILES

File Name	Description
Catalogue File	This is a flat file containing all the records of the stars. (Around 100,000 entries for the Hipparcos catalogue and around 1,000,000 entries for the Tycho catalogue)
Index File	This file contains the R-Tree index which is used for instantaneous access.
Searched File	This file contains the catalogue record numbers of all stars that lie within the window specified by the user.
Quantised File	This file is a Grid File containing the bitmap which represents the converted feature space.
Filtered File	This file is a Grid File containing the bitmap which represents the transformed feature space, i.e. the result of Wavelet Transform applied to the Quantised File.
Connected File	This file is a Grid File containing the bitmap in which each cluster in the area of interest is uniquely labelled. It is the result of Connected Component Algorithm applied to the Filtered File.
Isolated File	This file contains information about the detected clusters. It can also be stored on disk for later retrieval and analysis.

LIST OF REFERENCES

1. A. Tenebaum & M. Augenstein
Data Structures Using Pascal (Second Edition)
Prentice Hall India
2. A. Silberschatz, H. Korth & S. Sudharshan
Database System Concepts (Third Edition)
McGraw Hill
3. J. W. Han & M. Kamber
Data Mining: Concepts and Techniques
Morgan Kauffman Publishers(2001)
4. Ivor Horton
Beginning Java 2 (JDK 1.3 Edition)
Wrox Publications
5. Patrick Naughton & Herbert Schildt
The Complete Reference Java 2 (Third Edition)
Tata McGraw Hill Publications
6. Antonm Guttman
R-Trees – A Dynamic index structure for spatial searching
7. Jung-Me Park & Hui-C huan Chen
Fast Connected Component Labeling Algorithm Using a Divide and Conquer Technique
8. G. Shikholeslami, S.Chatterjee & A Zhang
Wave Cluster: A Multi-Resolution Clustering Approach for very large spatial databases
9. T. Zhang, M.Livny, R.Ramakrishnan
BIRCH: An Efficient Data Clustering Method For Very Large Databases
10. G. Karypis, E.Han and V. Kumar
Chameleon: A Hierarchical Clustering Algorithm Using Dynamic Modelling

11. <http://www.sdss.org>
Sloan Digital Sky Survey
12. <http://www.iucaa.ernet.in>
Inter University Center for Astronomy & Astrophysics Web Site
13. <http://astro.estec.esa.nl/Hipparcos/catalog.html>
Hipparcos Space Astrometry Mission
14. <http://www.hypermaths.org/quadibloc/maps/mps0401.htm>
Sinusoidal Projections for Galactic Plot
15. <http://violet.pha.jhu.edu/support/tools/eqtocal.html>
Equatorial to Galactic
16. <http://www.astro.washington.edu/labs/clearinghouse/homeworks/hrdiagram>
H-R Diagram
17. <http://skyserver.fnal.gov/en/proj/advanced/hr/>
The Hertzsprung – Russell Diagram
18. <http://www.shef.ac.uk/uni/academic/N-Q/phys/teaching/phy103/hrcluster.html> *Open Cluster H-R*
19. <http://www.java.sun.com>
Java Programming Reference
20. <http://www.oreilly.com/catalog/jenut2/>
Java Online Examples

APPENDIX

Spectral Classes

The various spectral classes are summarised in the table given below:

Temperature (Kelvin)	Spectral Class	Colour of Star
35000	O	Blue-White
21000	B	Blue-White
10000	A	White
7000	F	Yellow-White
6000	G	Yellow
4000	K	Orange
3000	M	Red