Statistical analysis of a massive astronomical cross-matching process

by

Daniel Nuño

in the

Escola Tècnica Superior d’Enginyeria de Telecomunicació de Barcelona
Departament de Física

Advisor: Enrique García–Berro Montilla
Co-advisor: Marcial Clotet Altarriba

April 2016
Acknowledgements

Este proyecto simboliza el cierre de una etapa que ha sido muy importante para mí y de la que siempre tendré grandes recuerdos. También ha sido la primera vez que no he tenido claro a donde iba, o mejor dicho, a donde quería ir. Seguramente por eso he invertido tanto tiempo pensando en qué hacer, en vez de hacerlo.

Pese a todo, ahora miro atrás y todo lo recorrido no me parece poco. Y sé que, en gran parte, se lo debo a todos los que me han rodeado este tiempo. Primero, a mis tutores Marcial Clotet, Enrique García–Berro y Jordi Portell, por haberme dado la oportunidad de hacer un proyecto como este con el que tanto he aprendido. Nada habría dejado “un lacito” mejor al final de esta etapa. Pero, sobre todo, por su increíble paciencia, apoyo y guía, y haberme hecho un hueco entre ellos. Quiero agradecer en especial el apoyo de Marcial, a quien seguro que le he quitado unas cuantas horas libres.

Mucho tengo que agradecer también a los amigos que me han acompañado, Kamila, Juanra, Carlitos (que atrás quedan ya esas clases de Microondas), Judith, Pere,... y a todo el grupo de NDOML. Solo puedo sentirme orgulloso de teneros a mi lado.

Pero sin duda, mi mayor agradecimiento es a Manel. Contigo aprendí que no hay situación mala, solo punto de vista equivocado. Y sobre todo, que paset a paset, todo es posible.

No puedo olvidar a mis padres y hermanos, quienes siempre han estado ahí para darme su apoyo. Aunque Jordi dejó el listón muy alto con su proyecto, ¡espero estar a la altura!

Dani
Contents

List of Figures vii
List of Tables xi
Acronyms xiii

1 Introduction 1
   1.1 The Gaia mission ......................................................... 1
   1.2 Gaia data processing overview ...................................... 3
   1.3 Motivation ................................................................. 5
   1.4 Structure and plan of this work .................................... 5

2 IDU Cross-Match 7
   2.1 Obs-Src Match ............................................................ 8
   2.2 Sky Partitioner ............................................................ 9
   2.3 Match resolver ............................................................ 10

3 Supervised classification algorithm 13
   3.1 Algorithm structure ...................................................... 14
   3.2 Feature extraction ....................................................... 15
   3.3 Classification systems .................................................. 15
      3.3.1 Principal Component Analysis .................................... 17
      3.3.2 Linear Discriminant Analysis .................................... 17
   3.4 Decision system ......................................................... 18
      3.4.1 Bayes ............................................................... 19
      3.4.2 Mahalanobis distance ............................................. 20

4 Approach and implementation 23
   4.1 Features extraction ..................................................... 23
      4.1.1 General features ................................................... 24
      4.1.2 Shape features ..................................................... 24
      4.1.3 Motion features ................................................... 28
      4.1.4 Implementation .................................................... 31
   4.2 Classification system ................................................... 33
      4.2.1 Method selection .................................................. 33
      4.2.2 Implementation ..................................................... 35
   4.3 Decision system ........................................................ 37
### 4.3.1 Method selection

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method selection</td>
<td>37</td>
</tr>
<tr>
<td>Implementation</td>
<td>38</td>
</tr>
</tbody>
</table>

### 5 Results and analysis

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature testing</td>
<td>43</td>
</tr>
<tr>
<td>Magnitude variance testing</td>
<td>44</td>
</tr>
<tr>
<td>Shape related testing</td>
<td>44</td>
</tr>
<tr>
<td>Motion related testing</td>
<td>51</td>
</tr>
<tr>
<td>Training and recognition testing</td>
<td>54</td>
</tr>
<tr>
<td>MCG datasets for testing</td>
<td>55</td>
</tr>
<tr>
<td>Training testing</td>
<td>57</td>
</tr>
<tr>
<td>Recognition testing</td>
<td>59</td>
</tr>
<tr>
<td>Results analysis</td>
<td>59</td>
</tr>
<tr>
<td>Considerations on the configurable parameters</td>
<td>61</td>
</tr>
<tr>
<td>Classification results</td>
<td>64</td>
</tr>
<tr>
<td>Simulated data</td>
<td>64</td>
</tr>
<tr>
<td>Real data</td>
<td>69</td>
</tr>
<tr>
<td>Computational load</td>
<td>75</td>
</tr>
</tbody>
</table>

### 6 Conclusions

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conclusions</td>
<td>77</td>
</tr>
<tr>
<td>Future work</td>
<td>78</td>
</tr>
</tbody>
</table>

### Bibliography

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bibliography</td>
<td>79</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>The journey of Gaia to its operational orbit.</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Simulation of the average number of observations done by Gaia during 6 months, in Galactic coordinates.</td>
<td>3</td>
</tr>
<tr>
<td>1.3</td>
<td>Overview of the Gaia DPAC.</td>
<td>4</td>
</tr>
<tr>
<td>2.1</td>
<td>Match Candidate examples.</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Example of a Match Candidate Group</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>UML diagram of a Match Candidate Group</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>Diagram of the supervised classification system.</td>
<td>14</td>
</tr>
<tr>
<td>3.2</td>
<td>Top panel: example of an original data set before to a Mahalanobis transformation. Bottom panel: distribution of the data set normalized to unit-covariance. The ellipse and the circle represent the original and transformed data covariance, respectively.</td>
<td>21</td>
</tr>
<tr>
<td>4.1</td>
<td>Circular diagram examples. The top panel displays a circular diagram with 4 radial divisions and 12 angular divisions, while the bottom panel shows the same diagram with 5 radial divisions and 12 angular divisions.</td>
<td>25</td>
</tr>
<tr>
<td>4.2</td>
<td>An example of the radial mean percentage feature.</td>
<td>26</td>
</tr>
<tr>
<td>4.3</td>
<td>An example of the radial variance feature.</td>
<td>26</td>
</tr>
<tr>
<td>4.4</td>
<td>Top panel: observation map used to illustrate the concept of the radial linearity feature for a binary system. Bottom panel: circular diagram corresponding to the previous MCG.</td>
<td>27</td>
</tr>
<tr>
<td>4.5</td>
<td>Cluster counter example for a MCG with 5 sources.</td>
<td>29</td>
</tr>
<tr>
<td>4.6</td>
<td>Circular diagrams to illustrate the cluster counter feature for a MCG with 5 sources. The top panel shows the cluster counter feature with 4 radial and 8 angular divisions, while the bottom panel shows it with 5 radial and 12 angular divisions.</td>
<td>30</td>
</tr>
<tr>
<td>4.7</td>
<td>MCG partition by time.</td>
<td>31</td>
</tr>
<tr>
<td>4.8</td>
<td>Observations (top panel) and circular diagram (bottom panel) for a star with low proper motion.</td>
<td>32</td>
</tr>
<tr>
<td>4.9</td>
<td>Single MCG structure.</td>
<td>33</td>
</tr>
<tr>
<td>4.10</td>
<td>McgStatistic UML diagram.</td>
<td>33</td>
</tr>
<tr>
<td>4.11</td>
<td>McgFeature UML diagram.</td>
<td>34</td>
</tr>
<tr>
<td>4.12</td>
<td>Illustration of the better classification capability of PCA over LDA. $D_{PCA}$ and $D_{LDA}$ represent the decision thresholds.</td>
<td>35</td>
</tr>
<tr>
<td>4.13</td>
<td>LDA and PCA results for a dataset with two Gaussian distributions.</td>
<td>36</td>
</tr>
<tr>
<td>4.14</td>
<td>Structure of the training class implemented in our classification algorithm.</td>
<td>37</td>
</tr>
</tbody>
</table>
4.15 Structure of the classification (or recognition) classes implemented in this work.  
4.16 Mahalanobis versus euclidean distance.  
4.17 3D illustration of the Mahalanobis versus euclidean distances.  

5.1 Top panel: first shape-related configuration. Bottom panel: corresponding circular diagram.  
5.2 Top panel: second shape-related configuration. Bottom panel: corresponding circular diagram. In this MCG the dispersion in magnitudes is smaller than the distance between the MCGs.  
5.3 Top panel: second shape-related configuration. Bottom panel: corresponding circular diagram. In this MCG the dispersion in magnitudes is comparable to the distance between the MCGs.  
5.4 Top panel: third shape-related configuration. Bottom panel: corresponding circular diagram. In this case the proper motion is 1,000 mas/yr.  
5.5 Top panel: third shape-related configuration. Bottom panel: corresponding circular diagram. In this case the proper motion is 3,000 mas/yr.  
5.6 Evolution plot of the observations for the first MGC of the first configuration. The top left panel shows the entire set of MCG observations. The successive panels display clockwise the first, second and third fractions of the MCG observations.  
5.7 Evolution plot of the observations for the second MGC of the first configuration. The top left panel shows the entire set of MCG observations. The successive panels display clockwise the first, second and third fractions of the MCG observations.  
5.8 Evolution plot of the observations for the first MGC of the second configuration. The top left panel shows the entire set of MCG observations. The successive panels display clockwise the first, second and third fractions of the MCG observations.  
5.9 Evolution plot of the observations for the second MGC of the second configuration. The top left panel shows the entire set of MCG observations. The successive panels display clockwise the first, second and third fractions of the MCG observations.  
5.10 Circular MCG from dataset 1.  
5.11 Two-sources MCG from dataset 1.  
5.12 High proper motion MCG from dataset 1.  
5.13 Three sources MCG from Dataset 2.  
5.14 Five sources MCG from Dataset 2.  
5.15 Dataset 1 MCG projections.  
5.16 Dataset 2 MCG projections.  
5.17 Dataset 3 MCG projections.  
5.18 Wrongly detected circular single source.  
5.19 Wrongly detected two source MCG.  
5.20 Wrongly detected triple source MCGs.  
5.21 Trained galaxy.  
5.22 Trained nebula.
5.23 Some examples of spurious detections in real data. The top panel shows spurious detections around a source, while the bottom panel displays an isolated spurious detection.

5.24 Two detected nebulae.
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Main parameters of the Gaia instrumentation.</td>
<td>3</td>
</tr>
<tr>
<td>5.1</td>
<td>Configuration for dataset 1.</td>
<td>55</td>
</tr>
<tr>
<td>5.2</td>
<td>Configuration for dataset 2.</td>
<td>57</td>
</tr>
<tr>
<td>5.3</td>
<td>Dataset 1 results.</td>
<td>60</td>
</tr>
<tr>
<td>5.4</td>
<td>Dataset 2 results.</td>
<td>60</td>
</tr>
<tr>
<td>5.5</td>
<td>Dataset 3 results.</td>
<td>61</td>
</tr>
<tr>
<td>5.6</td>
<td>Ratios of MCG correctly classified for dataset 1.</td>
<td>63</td>
</tr>
<tr>
<td>5.7</td>
<td>Ratios of MCG correctly classified for dataset 2.</td>
<td>64</td>
</tr>
<tr>
<td>5.8</td>
<td>Results of the classification distributions.</td>
<td>68</td>
</tr>
<tr>
<td>5.9</td>
<td>One source MCGs simulated parameters.</td>
<td>69</td>
</tr>
<tr>
<td>5.10</td>
<td>Two source MCGs simulated parameters.</td>
<td>69</td>
</tr>
<tr>
<td>5.11</td>
<td>Classification results on real data.</td>
<td>71</td>
</tr>
<tr>
<td>5.12</td>
<td>Classification results for all the Gaia data available so far.</td>
<td>75</td>
</tr>
</tbody>
</table>
Acronyms

A
AC Across-Scan. 2
AGIS Astrometric Global Iterative Solution. 7
AL Along-Scan. 2

C
CCD Charge Coupled Devices. 2
CU Coordination Unit. 4

D
DPAC Data Analysis and Processing Consortium. 4, 7
DPC Data Processing Center. 4

E
ESA European Space Agency. 1

F
FLOP Floating-point Operations. 4

H
HIPPARCOS High Precision Parallax Collecting Satellite. 1
HPM High Proper Motion. 7, 28, 29, 44, 49, 51, 52, 54, 55, 61, 62, 64, 65, 67

I
IDT Initial Data Treatment. 1, 7
**IDU** Intermediate Data Updating. [4, 7]

**L**

**LDA** Linear Discriminant Analysis. [15, 18, 33, 35, 53, 67]

**LSF-PSF** Line/Point Spread Function. [7]

**M**

**MC** Match Candidate. [8, 9]

**MCG** Match Candidate Group. [vi, 3, 9, 10, 15, 17, 23, 25, 27, 32, 35, 40, 43, 49, 51, 69, 71, 75]

**P**

**PCA** Principal Component Analysis. [13, 16, 17, 33, 35]

**PhotPipe** Photometric Pipeline. [7]

**S**

**SMCG** Singular Match Candidate Group. [31, 32]

**T**

**TB** Terabyte. [4]

**TDI** Time Delayed Integration. [2]

**TEC** Total Error Classification. [19, 20]

**X**

**XM** Cross Match. [4, 7, 9, 10]
Chapter 1

Introduction

1.1 The Gaia mission

In August 1989, more than 2000 years after Hipparchus, the European Space Agency (ESA) launched High Precision Parallax Collecting Satellite (HIPPARCOS), the first astrometric satellite. HIPPARCOS collected scientific data between 1989 and 1993, and the resulting Hipparcos Catalogue was published in 1997. This catalogue contains positions, distances and proper motions for about 120,000 stars. Furthermore, these measurements are 200 times more accurate than any previous determination. Its successor, Gaia, is the most ambitious astrometric space mission up to date. Adopted within the scientific programme of ESA in October 2000, it was launched in December 2013. Gaia is measuring with very high accuracy the positions and velocities of a large number of stars, galactic and extragalactic objects. Consequently, a detailed three-dimensional map of more than 1 billion stars will be obtained, including most of the objects up to the 20th magnitude and even beyond. The precision of the angular (parallax) measurements will be of about 20 μ arc sec at the 15th magnitude.

Gaia continuously scans the sky using the proven principles of the HIPPARCOS mission. As shown in Fig. 1.1 it orbits around the L2 Lagrangian Point and will provide a complete survey down to the limit of the sensitivities of its instruments. Full sky coverage is possible as the satellite spins around its own axis, which itself precesses at a fixed angle of 45 degrees with respect to the Sun.

On average the satellite will observe each star about 85 times during the 5 years of nominal mission. For each of these observations (called transits) it will record its brightness, colour and, most importantly, its position. Fig. 1.2 is a Mollweide projection of the sky showing the average number of observations detected and observed by Gaia during
6 months. In this figure the scanning law of Gaia combined with a simulated model of the Galaxy can be clearly appreciated. The artifacts produced by the scanning law will disappear as the mission progresses. It also shows how Gaia will scan the Galactic plane more times than other regions.

Gaia combines information from the several instruments available on-board. The data includes astrometric, photometric and spectroscopic measurements. The satellite has two telescopes which are combined onto a single focal plane composed of 103 state-of-the-art Charge Coupled Devices (CCD) devices, allowing each observation to be composed of 12 CCD transits (which we call observations). Table 1.1 summarizes the most important features of the focal plane of Gaia. In this table we use the Along-Scan (AL) and Across-Scan (AC) terms to refer to the width and height of the instrument.

Continuous measurement of stellar sources using CCDs implies a special operation mode, different than that of the typical shutter-based imaging, called Time Delayed Integration (TDI). It is based on a continuous charge shift from one pixel row to the next, synchronized with the satellite spin motion. It is done in each CCD therefore accumulating the charge during its corresponding integration period. In this way, long exposure times can be achieved with minimal distortion and blur. The total transit time of a stellar source
Chapter 1. Introduction

3

Figure 1.2: Simulation of the average number of observations done by Gaia during 6 months, in Galactic coordinates.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Focal Plane width</td>
<td>104.26 cm, 1.58°</td>
</tr>
<tr>
<td>Focal Plane height</td>
<td>42.35 cm, 0.69°</td>
</tr>
<tr>
<td>Image motion</td>
<td>60 arcsec/s</td>
</tr>
<tr>
<td>CCD Dimensions (AL×AC)</td>
<td>4500×1966 pixels</td>
</tr>
<tr>
<td>Pixel Dimensions (AL×AC)</td>
<td>10×30 µm</td>
</tr>
<tr>
<td>Typical Window Size (AL×AC)</td>
<td>12×12 pixels</td>
</tr>
<tr>
<td>Detection Window Size (AL×AC)</td>
<td>80×12 pixels</td>
</tr>
<tr>
<td>Fields of View basic angle</td>
<td>106.5°</td>
</tr>
<tr>
<td>TDI Period (AL pixel size)</td>
<td>0.9828 ms</td>
</tr>
</tbody>
</table>

Table 1.1: Main parameters of the Gaia instrumentation.

on the focal plane is about 102 s, during which 9 astrometric measurements (each with an effective exposure time of typically 4.42 s) are performed.

1.2 Gaia data processing overview

Given the nature of the Gaia mission, a huge amount of complex, extremely precise data, representing the multiple observations of a billion diverse objects, will be acquired and downloaded to Earth. Therefore, data processing constitutes a challenging task in terms of expertise, effort and dedicated computing power. Gaia will produce some
100 Terabyte (TB) of raw data. To deal with this huge amount of data it is estimated that a processing power of over $10^{21}$ Floating-point Operations (FLOP) will be required.

A large pan-European team of expert scientists and software developers, from over 20 countries, known as Data Analysis and Processing Consortium (DPAC) is responsible for processing the data, with the final goal of producing the Gaia Catalogue. This consortium is sub-divided into nine smaller, specialized units known as Coordination Unit (CU), with each unit being assigned a unique set of data processing tasks. Fig. 1.3 shows the flow of data within DPAC and the roles of the various CUs and Data Processing Center (DPC) (locations of the latter in red).

Here we focus on CU3 (the Core Processing CU), which has to organize and implement the data processing pipelines all the way from receiving the raw telemetry down to the core astrometric solution. Within CU3 there are different systems with specialized tasks. The Initial Data Treatment (IDT) performs the initial processing of the data as soon as it is received from the satellite, including a preliminary Cross Match (XM) – matching observations to corresponding sources – and a first treatment of the image parameters. This is a daily process which has limited scope on the data range available due to the downlink priority scheme. Therefore its output data has a limited scientific quality. On the other hand, the Intermediate Data Updating (IDU) is a cyclic system that will reprocess the entire set of raw data using the updated calibrations obtained so far.
Chapter 1. Introduction

will provide higher coherence between all the scientific results and correct any errors or bad approximations from previous iterations. One of the key components of IDU is the XM, which provides the final Gaia source list. There are other systems available which implement all the data reduction functionalities, but these are out of the scope of this project. IDU is the most relevant one for this project and is described in further detail in Chapter 2.

1.3 Motivation

Due to the nature of the objects observed by Gaia, and as further detailed in Sect. 2.2, the IDU XM task generates isolated groups of observations not related with each other. These groups are called Match Candidate Group (MCG) and different types of objects are expected to produce groups with different properties. Understanding the properties of the different types of groups is key to a successful IDU XM resolution, detailed in Chapter 2. Therefore, a set of statistics or features that describe the properties of the groups must be defined. The performance of the proposed algorithm will be tested using using simulated data. This is very convenient as the MCG simulator provides the true values for the statistics. However, it should be taken into account that to produce reliable results the MCG simulator must be configured correctly to produce realistic scenarios. Therefore, the properties of the MCG obtained in operations must be known in order to extract the relevant configuration parameters for the simulator. Nevertheless, simulations are unable to reproduce all the scenarios that arise in operations in terms of complexity and number of realizations. Another reasonable approach to test and validate this stage is to group similar MCG and analyze the results. If the defined MCG features characterize the MCG properly, these should allow the classification into several different groups according to the requirements set by the XM algorithm developers. In summary, this project aims to provide a set of features that describe the MCG and implement a tailored classification system that allows their classification according to the requirements of the XM developers.

1.4 Structure and plan of this work

This report is organized as follows. First, in Chapter 2 we present the IDU XM problem and the challenges it poses, identifying the main features that must be treated. It is followed by Chapter 3 where we describe the overall structure of the algorithm to be implemented, composed of a feature extractor, a classification system, and a decision system. In this chapter we introduce as well the several types of algorithms available.
Next is Chapter 4 where we present the design and implementations chosen for our solution. Chapter 5 is devoted to discuss the tests performed, and the results obtained so far with our algorithm on realistically simulated IDU XM data and the preliminary results on operational data. Finally, in Chapter 6 we summarize our major findings, we present our conclusions and we elaborate on forthcoming work.
Chapter 2

IDU Cross-Match

IDU is the instrument calibration and data reduction system more demanding in terms of data volume and processing power of DPAC. IDU aims to provide an updated XM using the latest attitude, geometric calibration and source catalogue available. As mentioned previously, the purpose of the XM task is to link each observation to the corresponding source from the catalog. IDU also provides updated calibrations for bias, astrophysical background and instrument model, including Line/Point Spread Function (LSF-PSF). Finally, it also yields updated image parameters, namely, positions, fluxes and quality indicators.

Basically, IDU incorporates the astrometric solution from the Astrometric Global Iterative Solution (AGIS) resulting in an improved XM but also incorporates the photometric solution from the Photometric Pipeline (PhotPipe). This allows IDU to obtain improved image parameters. These improved results are the starting point for the next iterative reduction loop. That is, the next DPAC processing cycle. Without IDU, Gaia would not be able to provide the envisaged accuracies. Thus, it is key to get the optimum convergence of the iterative process on which all the data processing of the spacecraft is based. IDU is also responsible to provide a consistent XM for all downstream systems. The complexity of this task arises from the very large volume of data that has to be processed, specially in terms of number of records. To do this, IDU must merge and split sources when required, and also detect High Proper Motion (HPM) sources that might have been incorrectly matched in the daily processing of IDT. As IDU is a cyclic process, other problems or bugs detected in previous cycles can be corrected.

The IDU XM process treats all accumulated observations together, and can therefore perform a scientifically better processing than IDT. At the end of the mission, the number of detections is expected to be around $10^{11}$. Handling all this data in a single process is not a feasible approach. Therefore, the XM task is divided into three different
2.1 Obs-Src Match

In this first task, in order to compute the detection sky coordinates and obtain the preliminary source candidates for each individual detection the input observations are processed in a time-ordered way. The purpose of this first task is to identify, for each detection, all the possible matching sources from the Gaia catalogue, producing the so-called Match Candidate (MC). The MC contains the detection identifier and a list of candidate sources. A very simple example is displayed in Fig. 2.1. In this figure blue symbols are observations, red symbols are sources and grey lines are the links between observations and sources. In this example each MC only has one source candidate.

The first step of this task is to determine the sky coordinates. That is, the position of the detection in the sky. Once the observation sky coordinates are determined, a list of nearby sources is computed. This list is extracted from the input Gaia catalogue, which is the result of the previous processing cycle. These sources are propagated, using their
Chapter 2. *IDU Cross-Match*

parallax and proper motion, to the relevant epoch for each detection. The candidate sources are then selected based on a pure distance criterion. The final output of this first step is a set of MCs for the whole accumulated mission data.

### 2.2 Sky Partitioner

This second task groups the results from the previous step according to the source candidates provided for each individual detection. The objective is to determine spatially-isolated groups of detections, all located in a rather small and confined region of the sky. Therefore, this step does not perform any scientific processing but simply provides an efficient spatial data arrangement by solving region boundary issues and high proper motion scenarios. This stage only acts as a bridge between the first time-based and the final spatial-based processing.

The process starts by loading all MCs for a given region of the sky. Then, it takes the first MC and initializes a new Match Candidate Group (MCG). Afterwards, for each source listed in the MC it extracts the observations linked to this source and loads the corresponding MC for these detections. If these MCs are not already contained in the MCG it adds them to the group. This process finishes when all the detections connected by source candidate links are in a single group. This process is illustrated in Fig. 2.2 where three sources (in red) and their corresponding observations (in blue) can be seen. First, the algorithm includes in the group a single MC from one source. Then, it takes the source candidates from this MC and retrieves the rest of the MCs linked to it. After this, it scans these MCs looking for links to other sources. By doing so, it finds that some MCs are also linked to the other source. Then, it retrieves all MCs from the other source, and adds them to the group. The algorithm performs this process recursively until, when finished, it has effectively created a single MCG that contains all linked MCs that is, observations. Fig. 2.3 shows the UML diagram of a MCG where it can be seen how it is actually built upon the MC produced in the first Obs-Src Match task.

This algorithm is executed over all the input observations, i.e., MCs and the final result of this task is a set of MCGs. This set contains all the input observations organized in spatially-independent groups. There is a clear advantage from a processing point of view in looping through the MCG rather than the MCs. This process removes any boundary issues, so now each MCG can be processed independently from others, as they do not share any information. Therefore, understanding the nature and properties of these objects is crucial for the final XM resolution stage. These MCGs are the basic input to our project.
2.3 Match resolver

This is the final task where the XM is solved and the final data products are produced. This task is ultimately a spatial-based processing where all detections from a given isolated region of the sky, a MCG, are solved together taking into account all relevant observations and sources. The aim of this stage is to produce a Match table that contains, for each observation, the assigned source from the catalog. In some cases this will be a simple assignment, but there can be conflicts. These conflicts can be classified and labeled as follows. New sources: some observations will have no corresponding entry in the available Gaia catalogue. Therefore a new source must be created for these detections.
as all observations must belong to a source in the catalogue (there cannot be any orphan
detection). Source split: two observations might end up linked to the same source after
the previous tasks. However, if these have been detected at very similar times on-board
Gaia it means that there must be two different sources instead of just one. A clear
example of this is a binary star that has been resolved on-board but that previously
was a single source in the catalogue. In this case, the existing source is split into two.
Finally, the last possible conflict is Source merging: it happens when multiple catalogue
sources are linked to the same group of detections. If the solving algorithm finds that
these sources only link to the same observations it will merge them into a single source.
The details of this task, although the most scientifically relevant, are out of the scope
of this project. The actual solving process is not relevant for the improvements to be
developed here.
Chapter 3

Supervised classification algorithm

Traditional classification systems, such as discriminant analysis, have been known and used since the 1930s [1]. These systems are widely used and their performance is comparable, when properly implemented, to the latest machine learning algorithms. One of the main advantages of these algorithms is that they are easy to test because their execution is always stable and deterministic. They can be divided in two main families, Linear and Non Linear, depending on whether the boundaries between classes are linear or not. Furthermore, some algorithms do not require previous training, while others require a previous calibration in order to classify the data. Self-trained algorithms dynamically change as new data enters and this might lead to unexpected or uncontrolled situations that cannot be anticipated beforehand. Therefore, in some cases it is preferable to train the algorithm with known reference data and then apply this to the bulk of input data.

Previous research analyzing the performance of different classification systems shows that the best approach depends mostly on the specific dataset. Some tests comparing machine learning, neural networks, and statistical analysis show that any of them might deliver the best performance depending on the datasets being treated [2]. Actually, in most cases the data is the performance constraint of the system, and not the method itself. Therefore, it is important to process appropriately the data so the objects are representative without flooding the system with redundant information. Refs. [3] and [4] also demonstrate that a good pre-processing of the data can enhance the system performance, discarding irrelevant features while keeping those relevant for the selection systems, like AdaBoost or Principal Component Analysis (PCA). We selected the discriminant analysis as a good candidate to implement our classification system. Further details on the analysis and selection of the discriminant system can be found in Sect. 3.3.
3.1 Algorithm structure

We have chosen an otherwise typical three-phase classification scheme. This division also provides versatility in the implementation if changes are required in any phase. It is also a good coding practice. In particular, the definition of a common interface for each phase allows different implementations for them without modifying the others. Therefore, different approaches for any phase can be tested and compared easily. This is a convenient approach to test different configurations as these can change the performance of the overall discrimination system.

The three phases are: **Feature extraction**, **Classifier system** and **Decision system**. The first two are mainly focused on information extraction and pattern recognition, while the last one implements the decision system guided by the previous outputs and the configuration. Fig. 3.1 shows a diagram of the three phases.

The feature extraction phase transforms the original input data to a more suitable form for our purposes. The aim is to extract the relevant information so that pattern recognition can be performed efficiently. This part is crucial as if done incorrectly it can limit the performance of the whole system. After being trained the classifier system will process the extracted features, providing a multidimensional representation of the object specifically transformed for our purposes, in order to facilitate the classification decision and minimizing the error probability. Finally, based on the outputs of the classifier and the configuration parameters, the decision system performs the actual classification of an object into a given group. In fact, the two first phases can be considered as a mere preprocessing stage for this final process. In this project, different decision systems have
been implemented and compared. In what follows we describe each one of these phases in more detail.

### 3.2 Feature extraction

Due to the structure of the data, a direct classification analysis would not be efficient (neither in computation time nor in error probability). The relevant information for classification has to be extracted first.

The input data to our processing are the MCGs as defined in Sect. 2.2. These objects provide detailed information about the actual observations. However, they do not have information about their properties as a group, such as shape or motion. In the way they are defined, two MCGs with very similar shapes could be very different in terms of their definition (observations, sources, links and attributes).

In order to perform a good classification, the features that characterize the MCG must be extracted. These features will be the actual input to the discriminant system. For example, for the case of Linear Discriminant Analysis (LDA) features from objects of the same class are considered Gaussian distributed. Hence, distributions that do not have this shape may fail even if they are characteristic for a given class. To obtain features that are relevant for the classification, understanding the MCG data structure is of crucial importance. The structure stores the data as obtained by Gaia. Namely, it stores detections, sources, position, brightness, and so on and so forth. All this information is absolute data about specific observation. This is important for the Gaia data reduction systems, but we are more interested in a generic data description that defines the MCG features. Therefore, we will define a set of generic features that provide information for the whole group. We obtain them by analyzing the observations in a group as a whole. These features can be organized as general, shape-related and movement–related. They are discussed in Sect. 4.1.

### 3.3 Classification systems

Classification systems have been the subject of exhaustive studies and developments for a long time, and there are several different approaches. From the most recent machine learning algorithms, like Self Organizing Maps or Neural Networks, to the older and more generally used pattern recognition algorithms, like discriminant analysis.

The classification system aims to maximize the differences between objects of different classes, giving a set of values that characterize them. These values can be seen as features
of the objects, but they do not represent any geometrical or physical feature. Rather than this, they are like a score that will allow the decision system to put the object in one or another class. Therefore, the decision system only needs to compare these scores with the expected values for each class and decide to which it belongs. An important point about the classification system is that it needs to be trained. Training datasets can modify the performance of the system, so it is a crucial step. Good training datasets can be obtained in different ways. In our case, a first option would be to manually filter MCGs with some expected features by visually inspecting them. This is, manually classifying them into their expected class. With this data, a first training can be done and later run over the data. Then we could manually inspect the classification results, further filter the correct cases and train the system again. This is a slow, manual and prone to errors training system. Another solution is to use a MCG simulator to generate MCGs with specific features that we expect in our class. It allows us a better control of the features that are used for training. Then, these simulated MCGs are used to train the classification algorithm. Once trained, the algorithm can be used over real data. The only drawback of this method is that the MCG simulator might have some limitations that do not allow every possible feature combination.

As previously discussed in Sect. 3.2 MCGs may have an arbitrary number of different features that characterize them. Therefore, the analysis takes places in a \( n \)-dimensional space, as each feature is considered a dimension. All MCGs can be represented in this space. They will form clusters where the elements have similar properties, according to the features computed. If the features provide relevant enough information, the clusters will be disjoint and can be clearly selected separately. In theory, it should be possible to separate different MCGs in such \( n \)-dimensional space. However, in practice, the number of dimensions might be too large to allow a feasible selection in the original space. A usual way to bypass this limitation is to reduce the dimension of the selection space, therefore reducing the computing requirements while maintaining all the relevant information.

We have considered two population analysis methods that allow such an efficient \( n \)-dimensional analysis. Specifically, we used Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). Both methods, on which we elaborate in the forthcoming subsections, are viable options for our classification system. Later, in Sect. 4.2.1 we will compare both approaches and we will argue our choice for the one selected for this study. The detailed description of the system implemented here will be given in Chapter 4.
3.3.1 Principal Component Analysis

PCA is a statistical method to characterize objects in a reduced dimensional space while prevailing their distinctive information. The method computes a transformation that reduces the amount of data needed to define an object. In other words, it ignores the irrelevant information, keeping the combinations of the input information that hold the main characteristics of the objects.

Given an \( n \)-dimensional vector representation of an object (being these dimensions the features of the MCG), PCA can be used to find a subspace whose basis vectors correspond to the maximum-variance directions in the original space. This transformation makes all objects maximize their differences, therefore optimizing their representation, while dimensions with less variance (therefore with less object isolating capabilities) are ignored. As these maximum-variance axes are usually a combination of different axes of the original space, and some original axis can be omitted, this procedure is analogous to a dimensional reduction.

For the PCA training stage, the maximum variance for the training samples is searched, and the transformation needed is stored. Then, using this information, boundaries that can isolate the different classes are computed. The mean and dispersion of the classes are considered for the boundaries, trying to minimize the misclassification probability. For any object being classified through PCA, the transformation is applied (which means a linear operation on its features). The membership to a given class is obtained by looking at the region between boundaries where the transformed data belongs. Leaving orphan objects (not considering an object belonging to one or another class) is a significant problem in PCA. Making object differences more notable by spreading away the data is its only aim. The implications of this are discussed in Sect. 4.2.1.

3.3.2 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) also known as Fisher’s linear discriminant analysis, searches for the vectors in the original \( n \)-dimensional data space that best discriminate among classes (rather than those that best describe the data, as PCA does).

Given a number of independent features, LDA builds a linear combination of these which yields the largest mean differences in terms of Euclidean distance between the given classes. LDA operates as follows. First, the membership to a class is considered. Therefore, treating all objects of each class as a single object (more exactly, as Gaussian-shaped objects) and performing a PCA-like transformation is the first objective. This procedure, as with PCA, maximizes the differences between them. Secondly, an inverse
PCA transformation is done, therefore making all objects of each class as similar as possible. This is done aiming at objects of the same class to appear as far from boundaries as possible, therefore decreasing the classification error probability. A mathematical formulation of these operations is given by the following equations. We first define the between-class scatter matrix, which characterizes the spread of the classes in the original $n$-dimensional space:

$$S_b = \sum_{j=1}^{c} (\mu_j - \mu)(\mu_j - \mu)^T$$  \hspace{1cm} (3.1)

where $\mu_j$ is the mean feature-vector of class $j$, $c$ is the number of classes, and $\mu$ represents the mean feature-vector of all classes. We then define the within-class scatter matrix. This matrix characterizes the spread of the data for each class:

$$S_w = \sum_{j=1}^{c} \sum_{i=1}^{N_j} (x_{ij} - \mu_j)(x_{ij} - \mu_j)^T$$  \hspace{1cm} (3.2)

where $x_{ij}$ is the $i$-th sample of class $j$, and $N_j$ is the number of samples in class $j$.

The LDA projection is found by optimizing the relation between both. The goal is to maximize the between-class scatter while minimizing the within-class scatter. One way to do this is to maximize the ratio

$$\frac{\det S_b}{\det S_w}$$  \hspace{1cm} (3.3)

The transformation that maximizes the ratio is found by an eigenvector decomposition of matrix $S_w^{-1}S_b$, as long as $S_w$ is a non-singular matrix. Through this transformation, at most $c - 1$ (being $c$ the number of classes) eigenvectors of the decomposition will be non zero \footnote{4}. Therefore, it fixes an upper bound in the dimension of the final subspace of $c - 1$.

The classification capacity is the priority for this method, at the expense of object characterization. This results from the classes being distant while inner objects of each class being as concentrated as possible, therefore minimizing the objects near boundaries. Those objects are problematic due to the difficulty in classifying them, so this method is very useful in that sense.

### 3.4 Decision system

This section focuses on the different methods employed to assign classes to a projected object. Decision over non-transformed data is very inefficient in terms of computing time, and also presents problems in classification performance due to excess of irrelevant data. Excessive data behaves as noise, complicating the decision of assigning a class. Therefore,
all data is projected through the classification system before taking the decision. Two methods have been analyzed and implemented. The first of these is a Bayesian method, which is based on the probability of belonging to a class, whereas the second one is known as the Mahalanobis distance method, that is based on the subspace distance to the object statistical distributions. We describe them separately, in the next two subsections.

### 3.4.1 Bayes

This method is based on minimizing the **Total Error Classification (TEC)**. This is, to minimize the ratio of objects that are incorrectly classified. The Bayes method classifies an object to the class that has higher conditioned probability of belonging to it. That is, for a sample $x$ and $c$ possible classes, the Bayes rule will assign it to class $i$ if it satisfies the Bayes inequation:

$$P(i \mid x) > P(j \mid x) \forall j \neq i$$

Generally speaking, $P(i \mid x)$ is difficult to obtain, but $P(x \mid i)$, being this the probability of obtaining a sample $x$ from class $i$, can be obtained more easily. Nevertheless, another problem shows up. Clearly, $P(x \mid i)$ depends on the number of training samples used, and if they do not represent well enough the statistical distribution of the classes this probability will be wrong. To bypass this, we can assume a type of statistical distribution and derive from it $P(x \mid i)$ theoretically. Assuming that the distributions of classes are Gaussian, it is possible to obtain Eq. (3.5) where $C_i$ is the covariance matrix for class $i$:

$$P(x \mid i) = \left( \frac{1}{(2\pi)^{\frac{d}{2}}|C_i|^{\frac{1}{2}}} \right) e^{-\frac{1}{2}(x-\mu_i)^T C_i^{-1}(x-\mu_i)}$$

This method can run over the original samples or the projected ones, but using the original samples will cause a worse error ratio, since the projecting process maximizes the differences between covariances and means of the different classes.

By comparing the probability of belonging to two different classes in an inequation, a discriminant function can be obtained, given by Eq. (3.6), where $p_i$ is the *a priori* probability of any object belonging to class $i$, or in other words, the frequency of that class:

$$f_i = \mu_i C_i^{-1} x^T - \frac{1}{2} \mu_i C_i^{-1} \mu_i^T + \ln(p_i)$$

$$P(i \mid x) > P(j \mid x) \forall j \neq i$$

$$P(x \mid i) = \left( \frac{1}{(2\pi)^{\frac{d}{2}}|C_i|^{\frac{1}{2}}} \right) e^{-\frac{1}{2}(x-\mu_i)^T C_i^{-1}(x-\mu_i)}$$

$$f_i = \mu_i C_i^{-1} x^T - \frac{1}{2} \mu_i C_i^{-1} \mu_i^T + \ln(p_i)$$
In our particular case this probability has been neglected, as the frequency of object
classes in the training samples is not related to the real frequency. However, this can be
changed through the configuration, and the probabilities can be loaded independently
of the training samples.

The discriminating function gives a weight as a function of the probability that a sample
from class $i$ would be equal to $x$. This is very different to what was originally intended,
namely to compute the probability of $x$ to belong to class $i$. This is a very good classifier
system as it minimizes the TEC. However, it must link all objects with a class. Even
knowing that Eq. (3.4) applies, it is impossible to know $P(i \mid x)$ neither the ratio
between different probabilities. Therefore is not possible to set an uncertainty margin.
Since in our case some objects simply do not fit in a class, meaning that they are
unrecognizable or belong to a non-trained class, their classification is expected to fail.
That is, the algorithm should be able to detect that these objects do not fit into any
trained class and just discard them. Therefore, it is preferable to isolate these objects in
some unclassified category, which would extremely minimize the TEC as the problematic
objects are skipped.

3.4.2 Mahalanobis distance

The Mahalanobis distance is a measure of the distance between a point $P$ and a distribu-
tion $D$. All axes are ponderated with the covariance matrix of the distribution.
Consequently this distance measures how far a point $P$ is from the mean of the distribu-
tion considering the spread of $D$ in any direction, and also the covariance between
variables. This algorithm is widely used in pattern recognition and other statistical
analyses when dealing with multivariate data. The Mahalanobis distance is computed
using the next equation:

$$D_M = \sqrt{(x - \mu)^T S^{-1} (x - \mu)}$$  \hspace{1cm} (3.7)

The Mahalanobis distance transforms the data so the result is uncorrelated, also leading
to unit variance in all axes. In other words, the resulting covariance matrix is equal to
the identity matrix. Geometrically, it can be seen as an axis transformation (so the data
becomes uncorrelated on this axis) and a scaling factor (so the variance of each axis
becomes unity). With the transformed data, the Euclidean distance of any point $P$ to
the center of the distribution $D$ matches the Mahalanobis distance between $P$ and $D$.

For each class, a different Mahalanobis distance is measured. The smaller distance is
obtained for the nearest (or the most similar) distribution. The main advantage of this
method is that a value proportional to the similarity of the object to the class is given.
Therefore, a measure of the goodness of the classification is provided by simply comparing the different distances. Consequently, objects that cannot be clearly classified can be skipped, avoiding classification errors and also allowing to manually analyze these particular cases. This largely decreases the error probability as those cases with more chances of being incorrectly classified are isolated and skipped.
In the top panel of Fig. 3.2 a simulated two-dimensional Gaussian distribution is shown, which is then transformed so that the Mahalanobis distance meets the euclidean distance (bottom panel). The Mahalanobis procedure finds the new axes where the data is uncorrelated, which are plotted in black. The ellipses indicate the values of the standard deviations of the data in the axes directions, being the red ellipse equal to unit standard deviation, and the blue one equal to a standard deviation of three. To complete the transformation, data is scaled to meet the unit standard deviation in all directions. This can be seen in the bottom panel of Fig. 3.2 where the transformed data is shown. The blue ellipse is the original standard deviation ellipse, now transformed into the red circle, highlighting the unit covariance in all directions. After this transformation, an Euclidean distance measure gives the Mahalanobis distance. As an example, data has been marked as $A$ and $B$ in both panels. There is a large difference in distance to the center of the distribution between $A$ and $B$ in the original data. However, after the transformation, $B$ is much closer.
Chapter 4

Approach and implementation

This chapter details the approach adopted for implementing a supervised classification algorithm. The aim is to build a system that, given a set of characteristics for each object, computes a representation that allows the classification into several predefined classes. As previously mentioned, the algorithm is divided into three different stages. Each stage follows a given interface that allows several implementations. The three stages are Feature extraction procedure, which is discussed in Sect. 4.1, the Classification system, which is explained in Sect. 4.2 and the Decision system, which is thoroughly examined in Sect. 4.3.

4.1 Features extraction

The first stage computes the features or properties of the object that will be used in subsequent stages. Here we describe the features considered in this project, which have been designed to provide relevant information on the MCG to the classification stage. The MCGs, as described in Sect. 2.2, are basically a container for the detection information provided by Gaia. This raw representation is not suitable for the classification process. Therefore, the first step is to process the raw data and compute the relevant features for each object.

In order to define the MCG characteristics, a statistical analysis has to be done. The statistics calculated from all the observations need to be specifically designed to highlight the features that differentiate one group from another. All features have been designed so that statistics do not get biased by any rotation or expected transformation on the data due to differences in how it was acquired. Therefore, an adequate symmetry transformation is applied when necessary. It is worth noting that some features make
use of the MCG center concept for their computation. Such center is determined from the mean of all MCG observations coordinates. Features are presented here in three groups: General, Shape related and Movement related.

4.1.1 General features

General features are the most basic ones. These indicate basic properties of the observations that compose a MCG or, in some cases, the variation of these properties. The general features implemented are the following:

- **Number of observations**: Number of observations linked to the MCG.
- **Number of sources**: Number of sources linked to the MCG.
- **Flatness**: The flatness, defined as the ratio between the major and minor axis, of the ellipsoid defined by the closest and farthest observations to the MCG center.
- **Radius**: Distance of the farthest observation to the MCG center.
- **Scans**: Number of times the area of the sky contained in the MCG has been scanned by Gaia.
- **Magnitude standard deviation**: Standard deviation of the magnitude of all observations linked to the MCG.

4.1.2 Shape features

The number of observations of MCG objects varies considerably, from very few observations to large groups in dense areas of the sky. A priori, the shape of each MCG is random. Extracting shape information that is useful for categorization implies that some simplifications must be done. Also, a simplified approach reduces the computational cost while still providing meaningful information. We defined a basic shape feature named Circular Diagram, from which we can extract other metrics such as the Radial mean percent, Radial Variance and Radial Linearity.

- **Circular diagram**: This feature is a circular grid with a configurable number of angular and radial divisions, centered on the MCG center. The radius is equal to the distance of the farthest observation to the MCG center. Each bin represents the number of observations that, given their relative position to the center, fall inside that bin. This data is stored in a matrix so it can be used to compute other
Figure 4.1 shows two examples of this feature represented by a diagram. Note that both diagrams have been computed for the same MCG but with a different number of angular and radial divisions.

- **Radial mean percentage:** This is a representation of the radial frequency of the observations. In other words, observations with the same distance from the center of the MCG will end up in the same ring. The motivation behind this feature is to represent how concentrated are the observations with respect to the MCG center.
Statistical analysis of a massive astronomical cross-matching process

Figure 4.2: An example of the radial mean percentage feature.

Figure 4.3: An example of the radial variance feature.

and relative to the farthest observation. An example of this feature is displayed in Fig. 4.2.

- **Radial variance**: This feature computes the variance in the number of observations for the cells (or bins) within the same radius. Therefore, a different value is computed for each radius. The aim is to discriminate the objects with angular symmetry, like point-like objects or circular extended sources, from other kind of
objects with elongated shapes. An example of this feature is shown in Fig. 4.3 where, for each radius, the number of observations in each ring is shown.

- **Radial linearity:** This is a way to quantify the elongation of a MCG. To compute it, we take the cell with the maximum number of observations, determine the direction of that cell with respect to the center, and finally consider all the cells
aligned with it. The ratio of observations contained in the selected bins versus the total observations gives the radial linearity. This feature is designed to complement two other features, Movement linearity and Movement variance, described below, in Sect. 4.1.3. Specifically, HPM sources can be detected by measuring the stability of the movement direction and amplitude. However, the motion in itself is better quantified with the radial linearity feature.

This feature was initially designed to help identifying stretched MCG due to HPM stars. However, it has also proved to be very useful to identify MCG with two different sources. In this case, observations will be concentrated in the areas considered by the radial linearity, producing values close to 1.0. HPM sources are expected to also be assigned a large value for this feature, although in practice they do not reach 1.0, as in the two sources case. Other cases will get random values, most of the time around 0.25, as the area computed is a fourth of the total. Fig. 4.4 illustrates a system with two sources that has a radial linearity of 1.0.

- **Cluster counter**: This feature computes the number of clusters found in a given circular diagram. A cluster is defined by adjacent cells containing less observations than a given minimum value. This minimum is defined as the 10% of the mean number of observations in cells with more than one observation. This value is designed to avoid considering as the same cluster different sources separated by just one cell. This is a general purpose shape feature, designed to identify MCG with more than one source. Note that cells connected diagonally are not identified as the same cluster. Fig. 4.5 shows a MCG with 5 sources reasonably separated compared to the error in the detections.

The circular diagram for this MCG is displayed in Fig. 4.6. The MCG has been analyzed with different number of divisions in the circular diagram. The detected clusters are highlighted in pink.

### 4.1.3 Motion features

We have defined as well some features to extract information about how observations change their position over time. The position of the observations is affected by the observational error, so the MCG is divided into several parts (according to the time of the observations) to minimize individual observational errors. The number of divisions can be configured. It has been used to study the best classification setup for both accuracy and computing time. Fig. 4.7 shows an example of how a MCG is divided. The
features related to motion that we consider here are Movement linearity and Movement variance, which are described next.

- **Movement linearity**: This feature is the result of the scalar product of the vectors of the movement between the centers of each part. Therefore, the resulting value varies between $-1.00$ and $+1.00$, and it provides information about the variation on the direction of motion. High proper motion stars are expected to give the highest values in this feature, while other objects will give lower values.

- **Movement variance**: This feature is a measure of the variance of the amplitude of the movements over the divided parts of the MCG. It behaves as a validation test of the previous feature, but it is also useful to detect extended objects. Some MCGs, specially the ones with fewer observations where noise has a significant effect, may yield a high movement linearity randomly. When the motion is not regular in direction or amplitude is an indication that the apparent motion is due to other reasons, e.g., an extended object or noise.

HPM sources are expected to have high values in both radial linearity and movement linearity. Nevertheless, not all combinations of values of these features are possible.
Figure 4.6: Circular diagrams to illustrate the cluster counter feature for a MCG with 5 sources. The top panel shows the cluster counter feature with 4 radial and 8 angular divisions, while the bottom panel shows it with 5 radial and 12 angular divisions.

Typical causes for one of the linearity features being high and the other low include a small number of observations (therefore very incomplete MCG), or sources with significant (yet still low) proper motion. Fig. 4.8 shows a MCG with a high movement linearity, about 0.95, but very low radial linearity, about 0.23. As the proper motion is small, the detection noise has a large impact on the distribution of the observations. The cells used to compute the feature have been highlighted.
4.1.4 Implementation

Data resulting from the feature extraction process needs to be accessible to the other stages. A new data structure has been defined, namely, the **Singular Match Candidate Group (SMCG)**, to allow direct access to the feature information and the actual **MCG**.

Fig. 4.9 shows the UML diagram of the **Singular Match Candidate Group (SMCG)**.

We have defined an interface, named **McgStatistic**, to compute the features. Fig. 4.10 shows the UML diagram for such interface. It defines a **compute** method, as well as a name, units and range for each histogram. This method, once implemented, has all the instructions needed to calculate the feature. When creating a new **SMCG**, the task will check the configuration, looking for the list of features to be computed. It will then create the corresponding **McgStatistic** objects dynamically, and will finally call the compute method for all of them.

Not all features produce the same type of output data. We can get integers or floating point values, and they can also be either a single scalar value or an array of values. In order to store the different feature extraction results, a class has been implemented, named **McgFeature**. This class is designed to store any type of feature along with the identifier and the type of result (single value or array). The **SMCG** class has an array of **McgFeature** objects, to store all features results that have been configured. Fig. 4.11 shows the UML diagram of **McgFeature**.

With all the features extending a common interface it has been possible to implement a generic method that computes the statistics that have been configured through Java **properties**. The tasks can load a given property with the given statistics and then
Figure 4.8: Observations (top panel) and circular diagram (bottom panel) for a star with low proper motion.

Compute the features for each MCG, storing them in the corresponding SMCG. This approach has proven to be very flexible. Adding new features, modifying an existing one or selecting which ones must be executed in a given moment is trivial with this setup. This design also makes the code better organized and easy to understand.
4.2 Classification system

4.2.1 Method selection

**PCA** is widely used for efficient object characterization as it performs dimension reduction while keeping the most representative information, also giving high discrimination capacity between objects. However, **PCA** does not take into account the classes of the objects for the training, which makes it less suitable for class classification. There are some exceptions to this, as shown later in this section, but they do not apply to this project. On the other hand, **LDA** is focused on classification. The object representation information is reduced but classification information is kept. The classification performance is enhanced, making all objects of the same class more similar between them, and at the same time more distinct to those from other classes. As the main goal of
this project is classification, without any interest on other information about the objects in any class, our best choice is {LDA}. However, there are two conditions affecting small training data sets that should be mentioned:

1. The within-class scatter matrix, $S_w$, needs to be non-singular. At least $n+c$ samples are required to guarantee it. This issue can be bypassed if a PCA is previously run on the data, projecting it over an intermediate subspace. Since such intermediate subspace will be dimensionally reduced, the $S'_w$ matrix will be non-singular with fewer samples. This method can also be used when $S_w$ is non-singular for other reasons.

2. When samples are not representative of their classes (because of small data sets), the {LDA} distribution of the class is inaccurate as well. Therefore, the algorithm is unable to set the projection that best discriminates between the classes. This issue is illustrated in Fig. 4.12, which shows two classes with normal distributions having only two training samples for each class. The corresponding projections and boundaries for {PCA} and {LDA} are also shown. Fortunately, in our case very large data sets are expected, so these cases are shown only for the sake of completeness.

In contrast with Fig. 4.12, Fig. 4.13 shows a data set composed of two Gaussian distributions along with the projecting axes of {PCA} and {LDA}. The figure shows the projected data as well. It is interesting to note the difference between both methods. Of course, this is an extreme case, as the data has been artificially generated so that the axis of the maximum variance is parallel to the boundaries between the classes. Therefore, {PCA} is expected to fail. However, it can be seen that, in the absence of classes, most of the information to differentiate one sample from another is still kept.

To summarize, {LDA} proves to have better discrimination capabilities. Also, the issues shown for {LDA} are not expected to appear in our case, as they specifically affect small datasets. On the other hand, the problem with {PCA} shown in Fig. 4.13 depends on the
Chapter 4. Approach and implementation

Figure 4.12: Illustration of the better classification capability of PCA over LDA. $D_{PCA}$ and $D_{LDA}$ represent the decision thresholds.

data feature distribution. Consequently, it can happen on any dataset, making PCA less desirable. Therefore, the classification system selected for our implementation is **Linear Discriminant Analysis (LDA)**.

### 4.2.2 Implementation

The LDA algorithm consists of two stages, training and projection. Both stages are never executed simultaneously, as training is the preparation before the projection. This two-stage structure is common in other classification systems. An interface for each stage has been defined to allow different implementations.

Fig. 4.14 shows the training interface and the implementation structure. A minimum of $c$ features vectors and labels must be used for training, being $c$ the number of classes. The training stage aims to find the appropriate transformation in order to classify the given input classes. To train the LDA system, a set of known MCG is selected for each class to be classified. MCG are stored in different directories depending on their class. This is used by the program to know which class is assigned to each MCG. The LDA performs two steps in order to find the appropriate projection. the first step is the so-called feature extraction. Within this step the configured features are computed for all MCG. This data is stored and accumulated, as all MCG objects are needed for the training. It follows, the projection step. Once all MCG have been processed and features have been extracted, the training process (detailed in Sect. 3.3.2) starts. This process finds the base of the projecting space where trained classes are most distinct from each other, therefore maximizing its classification capabilities.
After the training stage, a projection matrix is computed. The training dataset is projected to the resulting subspace and stored into another matrix. Both are stored in files, allowing the classification of MCG based on the training performed. The projected training dataset contains information about the distribution of the training classes in the projecting space. These are used by the decision algorithm to compute the boundaries between classes.

The projection stage transforms the data into the projected space where the classification is performed. The same features used in the training must be computed for the classification. Otherwise, the process will fail, as the dimensions of the feature vectors
and the projecting matrix will not match. The projection matrix obtained through the training is a linear transformation from the feature subspace to the decision subspace. This transformation is applied to the MCG feature vector to obtain the projected representation of the MCG used for the classification. Similarly as in the training stage, two steps are performed for each MCG in order to find its projection. The first of them is, again, the feature extraction. For the current MCG all specified features are computed. It follows the second step, which is the projection itself. The features are projected using the projection matrix obtained during the training. The projection is done for each MCG one by one. Once the projection is computed, we can move to the execution of the decision system as detailed in Sect. 4.3.2 below.

4.3 Decision system

4.3.1 Method selection

Three different decision methods have been implemented. The first of them is based on Bayes probability and the rest are based on distance to the projected classes distributions. For the Bayes method the probability of a MCG belonging to a given class is approximated by considering the distributions of the projected classes as Gaussians, and obtaining from its probability density function the class that best matches. The mathematical process was detailed in Sect. 3.4.1. This method is designed to obtain the minimum error rate when class distributions are Gaussian. However, it suffers from high error rates with non-Gaussian distributions. Within the distance method for each MCG we use a simple euclidean distance from the projected MCG to the center of all
classes to compute the distance to the nearest class. Finally, the advanced distance method measures the Mahalanobis distance (detailed in Sect. 3.4.2), which takes into account the standard deviation of the distributions in the different projected dimensions. This method has been designed to avoid classifying MCGs from unknown classes. A maximum Mahalanobis distance to the matched class distribution is used to validate the classification. As training classes are particular cases and do not cover all possible objects that belong to a non-trained class are detected by this method, simply skipping a classification if the distance is over a configurable value. However, trained classes can have intersected classification regions if they are similar enough. To avoid classification of MCGs that fall in this region, the ratio between the distances to the first and second nearest classes can be used as a reliability indicator. If the ratio is under a configurable value, the MCG is also skipped. Avoiding such classification decreases the error rate and improves the homogeneity of the resulting classes.

As said, the Bayes method is designed to have the lowest error rate for Gaussian distributions, but this is usually not our case. The simple distance method is forced to classify all MCGs and thus MCGs of unknown classes will still be classified — lowering the performance and thus leading to less homogeneous classification. Finally, the advanced distance method outperforms the simple distance as it takes into account the distribution of the classes rather than only their center. It is also capable of skipping some MCGs if they do not resemble any known class. Besides decreasing the classification error rate, it delivers more consistent classification results, as only MCGs that closely fit the trained classes are classified.

4.3.2 Implementation

As before, an interface has been defined so that different decision methods can be implemented. The UML class diagram of the interface can be seen in Fig. 4.15, which illustrates the basic operation of all decision systems. All the implementations have a common invoke method, which contains the algorithm to find the nearest class. The result is stored in a variable which can be retrieved by the get function defined. Following the mentioned interface, an intermediate class BaseRecognition is defined, from which all other decision classes inherit, implementing the common set and get functions. The steps to execute the decision system are the next two. First, we have the data input. Before running, all data needed must obviously be set. It always includes the projected objects and projected class centers, and may also need the class deviations depending on the method used. This is done through the implemented set functions. It follows the decision computation. Once the data is set, a call to the invoke method of the decision
Figure 4.15: Structure of the classification (or recognition) classes implemented in this work.

The system computes the class for the object. Once the decision has been done, the class can be retrieved through the appropriate get method.

The implementation and operation of each method is the following. For the case of the Bayes algorithm, we have that following the implementation of the interface, the method needs the centers and standard deviations of the trained classes to be set before executing, as they are needed to compute the discriminant function. This is done by computing them through the stored projected training dataset. Once the class distribution statistics are loaded, the invoke method computes the Bayes discriminant function — Eq. (4.1) — for all the trained classes. The most likely class is then stored and ready to be retrieved. The criteria followed by this method is given Eq. (4.1), which selects the class $i$ that maximizes the Bayes discriminant, being $\mu_i$ the mean vector of the class $i$, $x$ the projected MCG, $C$ the covariance matrix of all the trained MCG, and $p_i$ the frequency of class $i$.

$$
\max_i f_i = \mu_i C^{-1} x^T - \frac{1}{2} \mu_i C^{-1} \mu_i^T + \ln(p_i) \tag{4.1}
$$

Within the distance method the MCG to be classified and the class centers must be set, as class standard deviations are not used in this method. The invoke method implements an algorithm to compute the euclidean distance from the MCG projection to the center of each class. The closest class is then computed. The criteria followed by this method is given in Eq. (4.2), which selects the class $i$ that minimizes the expression, where $x$ is the projected MCG and $\mu_i$ the mean vector of class $i$. 

$$
\min_i d_i = \|x - \mu_i\|_2 \tag{4.2}
$$
Finally, we discuss the advanced distance method. The improvement introduced by this method requires the standard deviations of the projected classes. A function which computes the Mahalanobis distance from a point to a given distribution is implemented. Once all data is set, executing the invoke method recursively calls the Mahalanobis distance function, storing the distances to all the classes. Once the distances are computed, the reliability of the classification is tested by comparing the distances of the two nearest classes. Only if the distance to the nearest class and the ratio are within the configured value, the classification is done. The criteria followed by this method is given by Eq. (4.3), which selects the class $i$ that minimizes the expression, where $x$ is the projected MCG, $\mu_i$ the mean vector of class $i$ and $S_i$ the standard deviation matrix for class $i$.

\[
\min_i D_M = \sqrt{(x - \mu_i)^T S_i^{-1} (x - \mu_i)} \tag{4.3}
\]

To illustrate the differences between Mahalanobis and euclidean distances, Figs. 4.16 and 4.17 show two Gaussian distributions and a point between them. In this specific case, the euclidean distance yields the following results:

\[
D_1 = \sqrt{(x - \mu_i)^T (x - \mu_i)} = \sqrt{2.25} = 1.5
\]

\[
D_2 = \sqrt{(x - \mu_i)^T (x - \mu_i)} = \sqrt{7.25} = 2.69
\]

whereas the Mahalanobis distance yields the following results:
Figure 4.17: 3D illustration of the Mahalanobis versus euclidean distances.

\[
D_{M1} = \sqrt{(x - \mu_i)^T S_i^{-1} (x - \mu_i)} = \sqrt{18.41} = 2.86
\]

\[
D_{M2} = \sqrt{(x - \mu_i)^T S_i^{-1} (x - \mu_i)} = \sqrt{7.04} = 2.65
\]

Both distributions have the same standard deviation matrix, only differing in a rotation. Due to the axes of the standard deviations, the distribution with the most distant center to the point is, in fact, closer. Taking into account this effect largely improves the decision performance.
Chapter 5

Results and analysis

This chapter has been organized as follows. First, we check that the features extraction software implemented so far performs as expected, still under controlled conditions (Sect. 5.1). Next, we train the system with simulated data and verify that it is able to classify correctly the same data used for such training (Sect. 5.2.2). The simulated datasets used for this purpose are described in Sect. 5.2.1. Afterwards, we generate additional datasets (but still with the same configuration as those used for the system training) and verify that they are also classified correctly (Sect. 5.2.3). Finally, in Sect. 5.3 we perform two last tests that definitely evaluate the performance of the system. The first of these tests still uses simulated data, but without any constraints on the configuration or properties. The other test uses real data from the Gaia spacecraft. Both tests are assessed by visual inspection on the MCG shape versus the classification results. We also evaluate the computational load of the algorithm.

5.1 Feature testing

In Section 4.1, the MCG statistics that have been implemented were detailed. These statistics describe the properties that characterize the MCG and improve the classification. Therefore, they are crucial for the operation of the system. To ensure the correct operation of the designed features, a set of tests have been implemented to ensure that they operate as expected in a wide range of possible cases.

As detailed in Section 4.1, statistics are divided in groups: shape, motion and general. A collection of datasets has been implemented to test each group of features. These datasets contain MCGs with different simulated properties, allowing to check how the statistics perform for the different cases. As the results are stored, testing can be done
at any time changes are done in the code to ensure that the features keep operating correctly.

5.1.1 Magnitude variance testing

This feature does not belong to shape nor motion related statistics. Therefore, its testing is detailed independently. Two MCG configurations have been simulated to check its operation. The first configuration is composed of one MCG in which a high error in the magnitude measurement has been simulated. The error introduced is uniformly distributed between ±1.5 magnitudes for each observation. Comparing the result obtained with the theoretical value, the feature detection operates correctly. The second configuration is composed of one MCG in which a small error in the magnitude measurement has been simulated. The error introduced is uniformly distributed between ±0.5 magnitudes for each observation. The results for this case also match with the expected variance.

5.1.2 Shape related testing

The dataset used to test shape related features is designed to highlight different shape features. The dataset is composed of MCGs with three different configurations. The first configuration is composed of one MCG with one source. It has been selected because their observations are uniformly distributed in all angular directions. This should be highlighted by the radial variance and radial mean statistics. This configuration is shown in Fig. 5.1.

The second configuration is composed of two MCGs with two sources. The distance between sources is configured to generate two different possibilities. The first one, as shown in Fig. 5.2, is chosen to have a distance between sources far larger than the dispersion of observations around the sources. The second, shown in Fig. 5.3, has a distance between sources relatively similar to the dispersion of the observations. This difference causes the observations of each source to either fall in the same cell of the circular diagram, or to be distributed between two cells.

Finally, the last configuration is composed of two MCGs with HPM sources. These MCGs have been selected due to their elongated shape, as motion statistics are not computed for this dataset. The first MCG is selected as it is a problematic case. The simulated proper motion is of approximately 1,000 mas/yr, smaller than the second case. However, the main problem with this MCG is that the observations are aligned with the cell divisions in the circular diagram. The implications of this issue are discussed
further in this section with the test results. The second MCG is a HPM source with a proper motion of around 3,000 mas/yr, which is a rather large value. Therefore, radial linearity and radial variance statistics should detect high linearity and variance for both cases, but even higher for the second MCG.
5.1.2.1 Circular diagram tests

This tool is a simplification of the distribution of observations so other statistics can be efficiently computed while still providing meaningful information. This feature is easily tested by manual inspection. Comparing the observation plots and circular diagrams for each configuration of the dataset shows the correct operation of the feature.

5.1.2.2 Circular diagram linearity tests

This statistic uses the circular diagram tool to analyze the elongation of an MCG. It computes the ratio of observations aligned with the cell with maximum number of observations and the center. For more details see Sect. 4.1.2. No linearity detection is
Chapter 5. Results and analysis

Figure 5.3: Top panel: second shape-related configuration. Bottom panel: corresponding circular diagram. In this MCG the dispersion in magnitudes is comparable to the distance between the MCGs.

expected for the first configuration of our test, as its observations are distributed uniformly in all angular directions. As this feature has a region of interest equal to 2 divided by the number of angular divisions of the total area, the expected value for this feature with MCG like this is 0.25 (corresponding to 8 angular divisions). The result actually obtained in the tests is 0.24, therefore validating the operation. On the other hand, both the second and third configurations are expected to have large values of linearity. The
second configuration has two MCG with two sources. As the two sources will always be aligned with the MCG center, a 1.0 result is expected for both. The test result for both MCG is 1.0, and thus assesses the correct operation of this tool. The statistics results for the third configuration are not expected to be evident. As observations are distributed along the direction of motion, some of them may fall on neighbor cells, not aligned with the cell with the maximum observations. This specially applies for the inner radius level, where observations are expected to fall in all cells due to the width of the MCG. Also, the first MCG of this configuration is aligned with the division between cells. Therefore, about 50% of the observations fall outside the region computed. The results for this test are 0.5 and 0.97, which are consistent with the expected behavior.
Figure 5.5: Top panel: third shape-related configuration. Bottom panel: corresponding circular diagram. In this case the proper motion is 3,000 mas/yr.

5.1.2.3 Radial variance tests

This statistics represents the angular dispersion of observations for each radius. Therefore, it highlights whether the source is uniformly distributed on all angles from the MCG center, or whether observations are concentrated in some preferred axis. The first configuration has been specifically selected because of its uniformity. Therefore, a small variance is expected to be detected by the tool. The test results correspond with the expected and show variances between 1.5 and 3.5 (obs/cell)$^2$. On the other hand, the second and third configurations have all the observations concentrated in some regions. Therefore, larger values should be obtained. As the second configuration is the one with the observations more concentrated, it should be the configuration with larger values.
However, in radius levels with no observations, no variance should be detected. As expected, in the radius without observations the result is 0 \(\text{(obs/cell)}^2\). The first MCG has a variance in its last radius level of 1054 \(\text{(obs/cell)}^2\), while the second results shows 89 and 710 \(\text{(obs/cell)}^2\) for the penultimate and last radius levels. This corresponds with the results, with the first MCG having the higher value as it has the observations even more concentrated. In the third configuration, observations are distributed along the direction of motion. Therefore, large variances of the observations should be detected in all radius levels as observations are not uniformly distributed. However, variance on the number of observations is expected to be smaller in the inner radius level. Due to the width of the MCG in the across movement direction, observations can fall on all cells of this radius level. Such radius level is very important as the number of observations in these cells largely varies proportionally to the amount of proper motion the source has. The results show a variance in the inner radius level of 2.7 \(\text{(obs/cell)}^2\) for the first MCG while the second obtains 25 \(\text{(obs/cell)}^2\). The results for the other radius levels are between 5 and 15 \(\text{(obs/cell)}^2\). As expected, a large variance is obtained in all radius levels except for the inner level for the first MCG where the variance is smaller. Also, the second case has a larger variance than the first, specially in the inner radius, due to its high flattening.

5.1.2.4 Radial mean percent tests

This statistical tool determines how the observations are distributed along the radial direction. For the first configuration, a similar number for all radius levels should be given. However, two complications appeared. On one hand, the observations that fall on each radius depend on the area that they cover, which increases with the radius. On the other hand, observations are slightly more concentrated near the center. Thus, a blend of both features must show up in the results. Indeed, the results show 15%, 38%, 26% and 21%, from the inner to the outer results. This results are consistent with the outcome expected.

The second configuration is the only one expected to have non-uniform radial distribution. As the observations are concentrated around the sources, the inner radius levels are empty. In the first MCG the distance of the sources is far larger than the dispersion of the observations. Therefore, observations are isolated only in the last radius level, showing a result of 100% of observations in the last radius level. The second MCG sources are nearer, so some observations also fall on the penultimate radius level. Results are consistent with this, with a 26% of the observations falling in the penultimate radius level.
For the third configuration, observations are distributed along an axis. Therefore uniform values for all radius levels are expected, with small variations. A small increase (depending in the proper motion amount) in the first radius level is normal as observations can fall in all cells. A decrease is also normal for the last radius level. The results show observation means between 17% and 35% for four radius levels, which coincide with the expected values, showing an almost uniform distribution of the observations. However, the observations in the first radius level are higher for the second MCG (35%) than for the first (23%), which was no expected. This is due to width of the MCG being larger than one radius level, which makes observations in the center of the MCG fall also in the second radius level.

5.1.3 Motion related testing

The dataset used to test the motion related features consists of two different configurations of MCG with different properties that should be detected by the movement linearity or movement variance features, described in Sect. 4.1.3.

The first configuration is composed of two MCG. Noise has been added to the position of the observations of these MCG in different amounts. The first MCG is a HPM source with a proper motion of about 1,000 mas/yr (which is quite high) with some noise added. Fig. 5.6 shows the evolution of the observations, where some overlapping can be seen. Therefore, HPM source features should be detected. However, as the movement has been altered, irregularities should also be detected. This MCG is an important test as features should highlight this dual behavior.

For the second MCG of this first configuration, no proper motion has been simulated so its shape is only due to noise, despite being the most elongated. This can also be seen in the observations plot in Fig. 5.7.

The second configuration, illustrated in Figs. 5.8 and 5.9 is composed of two usual unaltered HPM MCG. However, to ensure the correct operation of the statistics, different values of proper motion are simulated. Therefore, the first MCG has a proper motion of about 1,000 mas/yr, while the second moves at about 3,000 mas/yr. The statistics should detect some difference between them, as the intrinsic noise in the position measure becomes less relevant as the proper motion increases.

As in the shape related case, motion-related statistics have been tested with this dataset. Its results, shown in the following subsections, are analyzed to ensure that the statistics operate as expected.
5.1.3.1 Movement linearity tests

This statistic computes the variability in the direction of the MCG movement. The MCG is divided in groups as a function of the observation time, and the scalar product of the difference between centers is computed. In the first configuration, noise has been added to the MCG motion. However, in the first case this noise is relatively small when compared with the size of the MCG, so the linearity of the movement can be
Chapter 5. Results and analysis

Figure 5.8: Evolution plot of the observations for the first MGC of the second configuration. The top left panel shows the entire set of MCG observations. The successive panels display clockwise the first, second and third fractions of the MCG observations.

large. Despite this, the noise should be detected by the movement variance statistic. The feature result for the first MCG shows a 0.9 of movement linearity. The second case for this configuration is an extreme case. Observations are randomly placed in any position of the MCG without any specific motion pattern. This should yield a movement linearity between $-1$ and $0$. As expected, the result is in this range ($-0.9$). For the second configuration, as both MCG are unaltered HPM, high movement linearity is expected for both. The results show a 0.97 movement linearity for the first MCG and 0.99 for the second one, as expected.

5.1.3.2 Movement variance tests

This statistic computes the variability on the amplitude of the MCG movement. As the units of this statistic are really small (milliarcseconds per nanosecond), and the mean results for different MCG types are very distant, this statistic is given in a logarithmic scale.

For the first configuration, a high variance is expected for both. However, the second case, as it is extremely noisy, should have an even larger variance. The first case is specifically designed for this statistic, as the noise added cannot be detected by the
Movement Linearity added. The results shows a variance of $-435 \log(\text{mas/ns})$ for the first case and $-410 \log(\text{mas/ns})$ for the second. The difference between the results ensures the feature gives enough resolution between slightly noisy MCG and extreme cases as the second. Also, the first case result is different enough in comparison with HPM sources to differentiate it. For the second configuration, low variance is expected for both, as the source motion is constant. The results show a variance of approximately $-455 \log(\text{mas/ns})$ for both, therefore validating the statistic operation.

### 5.2 Training and recognition testing

In this section, the tests run over the classification system (explained in Section 4.2) are specified. As with the feature testing (detailed in Section 5.1), this is designed to check the correct operation of the LDA system and detect any bug produced by changes in the code. Three tests have been implemented to check the operation of both the training and recognition methods. For each test a different dataset is designed. The datasets are designed so that the features that differentiate the classes are different, therefore testing the classification in different scenarios.
Chapter 5. *Results and analysis*

<table>
<thead>
<tr>
<th>Property</th>
<th>Circular MCG</th>
<th>2 source MCG</th>
<th>HPM MCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radius</td>
<td>100 - 1000 mas</td>
<td>100 - 1000 mas</td>
<td>100 - 1000 mas</td>
</tr>
<tr>
<td>Observations per source</td>
<td>70 - 80</td>
<td>70 - 80</td>
<td>70 - 80</td>
</tr>
<tr>
<td>Distance between sources</td>
<td>N/A</td>
<td>50 - 100 mas</td>
<td>N/A</td>
</tr>
<tr>
<td>Proper motion mean</td>
<td>0 mas/yr</td>
<td>0 mas/yr</td>
<td>1000 mas/yr</td>
</tr>
<tr>
<td>Proper motion variance</td>
<td>100 (mas/yr)$^2$</td>
<td>100 (mas/yr)$^2$</td>
<td>300 (mas/yr)$^2$</td>
</tr>
</tbody>
</table>

Table 5.1: Configuration for dataset 1.

### 5.2.1 MCG datasets for testing

Here we detail the datasets used to test the classification system and the decision system. Three different datasets have been designed. The tests classes being classified are different, therefore different features are required to be computed.

#### 5.2.1.1 Dataset 1

The first dataset implements three different classes to be classified. The classes have differences in shape and motion. Therefore, all features detailed on Section 4.1 are computed. The dataset classes simulated for this test are the following, with their features indicated in Table 5.1. Specifically, we used the following classes. First, we designed a circular MCG, which consists of a single source MCG. Fig. 5.10 shows an example of a MCG that belongs to this class. We also employed a MCG involving two sources, which is illustrated in Fig. 5.11. Finally, we employed as well a a HPM MCG with a simulated high proper motion. Fig. 5.12 shows an example of a MCG that belongs to this class.

Classes 1 and 2 can be distinguished by shape-related features. However, to adequately classify the 3 classes we require motion-related features, specially because in class 2 flatness and radial linearity are expected to be similar to those of the HPM MCG. This test checks that different classes can be classified even when their distinguishable characteristics are in different dimensions of the feature space.

#### 5.2.1.2 Dataset 2

Dataset 2 also implements three different classes. However, it focuses on shape classification as none of the classes have proper motion. Hence, no motion differences exist between classes. Features used to classify this dataset are the shape-related features: radial mean, radial variance and cluster counter, which are detailed in Sect. 4.1.2 and the general feature magnitude variance explained in Sect. 4.1.1. The classes simulated
for this dataset are the following, while the parameters are summarized in Table 5.2. First we use a two source MCG, the same illustrated previously in Fig. 5.11. A second MCG, made of three sources, is shown in Fig. 5.13. Finally, an extreme case is a MCG with five sources, shown in Fig. 5.14.

Class 1 can be easily distinguishable from the others. However, to classify the other two classes is not so obvious for the algorithm. The difference is in the number of sources. The shape of the resulting MCG remains the same, so these classes are more difficult to differentiate than in dataset 1. Also, the projection is computed in a more reduced feature sub-space, although the discriminating information is focused on this specific features.
Chapter 5. *Results and analysis*

5.2.1.3 Dataset 3

This dataset is designed only for the recognition testing. Its composition is the same of dataset 1. However, it has four times more MCGs than dataset 1. Recognition testing through datasets 1 and 2 have inputs for the classification that are the same than for the training. Therefore, the performance of the classification is expected to be better, as long as the trained classes are homogeneous and distinguishable with the configured features. Therefore, this test is implemented to check the classification with MCG belonging to the trained classes but that have not been used in the training. More information about the class details is specified in Sect. 5.2.1.1 for the dataset 1.

5.2.2 Training testing

Training tests consist on the feature analysis of the provided datasets and the computation of the corresponding projection matrix and projected dataset for the different classes. This process is explained in Sect. 3.3.2 and its implementation is detailed in
Reference data is provided to check the results, containing the projecting matrix, the projected dataset and its class labels. Therefore, comparing the computed results with the reference data ensures that the same dataset yields the same results.

Two tests are implemented. As the classes to categorize in each test are different, the features being used for the classification are also different. The tests results are not commented, as the projecting matrix obtained cannot be classified as reliable or
unreliable. Therefore, validating the reference data is done in the recognition tests, as the results can be easily checked for convenient classification. Once the recognition results are analyzed the reference data is validated and stored as reference.

### 5.2.3 Recognition testing

The recognition tests check the classification performance of the system. This includes both the projection (classification system) and the decision system, as the classification of a MCG requires both systems to operate correctly. For a set of MCG and a projection matrix, the test projects the input MCG and classifies it. The result of these recognition tests is a table of all the MCG analyzed and its detected class. As this classification is known, the results are compared with the reference results, validating the classification process if both results coincide.

Three tests are implemented. Two of them are the same tests implemented for the training testing detailed in Sect. 5.2.2. In this way, besides testing the recognition system, the procedure also contributes to a more consistent training testing, as the results of this test can be easily checked looking the resulting table of classification. The third test performs a classification of non-trained MCG.

### 5.2.4 Results analysis

Training results are difficult to analyze as their results are large matrices. However, they can be easily analyzed by checking the classification results. Therefore, the recognition results are analyzed to test the reliability of both systems. Therefore all the results of the datasets are analyzed with its recognition results.

Figures 5.15 to 5.17 show the projection for all MCGs for datasets 1 to 3. For a MCG to be classified, it must be inside the configured maximum distance, which is represented with the black ellipsoids. For these tests a three-sigma (standard deviation) maximum distance is selected. This parameter depends on how the training and the classified classes are. The more similar are all the MCGs of a class, the smaller this boundary can be. Also, if the training does not represent all the possible MCGs or if it is desired to classify similar objects that have not been trained, the maximum distance should be increased. For some classes, specially if these classes have similarities, its projections may be overlapped, as can be seen in Fig. 5.15 with the circular MCG and two sources MCG cases. In these cases, the relative distance between the two nearest classes act as a reliability indicator. If the ratio between the distances of the two nearest classes is within a configurable value, the classification is avoided. The red ellipsoid in Fig. 5.15
indicates where these two distances are equal. Tables 5.3 to 5.5 summarize the results for the three datasets.

We should consider some peculiarities of the projections. As can be seen, specially in Fig. 5.16 some patterns on the MCGs projections can be identified. This is specially true in the triple (red) and double (green) source MCG in comparison with other more homogeneous distributions. This can be considered as isolated groups inside the classification group, as they are clearly grouped. This shows that this system, besides classifying trained classes, can be used to detect new sub-classes. However, isolating these new classes can be difficult as the boundaries must be manually computed, because there
is no previous training. Nevertheless, it provides a useful tool. The information of the projected MCG can be used to check whether the trained groups can be divided or if they are homogeneous. In this case the patterns seen can be a product of the simulator not being enough random or homogeneous. However, the utility of manually checking the projections is proved as it can show additional information about the trained classes.

### 5.2.5 Considerations on the configurable parameters

In order to operate the algorithm, some values need to be configured. Here we explain how these values have been obtained. Also, some simulations used to ensure these parameters are optimum are explained.

We first discuss the MCG partition for motion related features. Motion related features split the MCG in groups to evaluate the movement between these groups. As the feature results may vary with different split values, the classification performance has been analyzed to find the appropriate number of splits per MCG. Two approaches have
been taken to evaluate this: the value of the motion features of a group of $\text{HPM MCG}$ and the classification and error rates of the system when executed with this values. On one hand, classification performance for different split values has been analyzed. The results show similar performance for split values from four to seven splits, while three and nine splits have slightly poorer performances.

Histograms of the movement linearity and movement variance features (as they are the only features affected by $\text{MCG}$ splitting) for different number of splits have been computed to see more clearly how the splitting affect in the feature computation. From the examination of these histograms the effect of the different number of splits can be assessed. The splitting can be seen as an average. Therefore, when a small number of splits is chosen motion information is ignored. However, as noise in the position acquisition has a significant effect, excessive splitting causes some $\text{MCG}$ to have values different than the expected results. For three splits, the features are found to lose considerable information. More than 95% of the $\text{MCGs}$ yield values of movement linearity between 0.95 and 1.00, which is not representative for all the variety of $\text{HPM MCGs}$. Nevertheless, eight and nine partitions show some $\text{MCGs}$ having negative movement linearity. Manual inspection of these $\text{MCGs}$ showed a smaller than average proper motion as the only cause of these results. Also, for each split value, the $\text{MCGs}$ with negative movement linearity are different. Therefore, a large number of splits affects in a different way each $\text{MCG}$. Consequently, taking into account this analysis, the optimum value is between four and seven, which confirms the results from the performance analysis.
Chapter 5. Results and analysis

Additionally, to evaluate how splitting affects to different MCGs, some have been individually analyzed. This analysis showed that some of the MCGs had a split value which makes them to obtain unexpected results in motion features. As the number of observations is not usually a multiple of the splitting parameter, the last group is composed of the last observations plus observations from the penultimate group, so all groups are equally large. The movement is averaged by the mean time of the group observations so the final result is not affected by the overlapping. However, if the last group has very few new observations, the noise in the position of these observations affects to this last motion split. The fewer the new observations of the last group, the more important the effect of noise is. Consequently, is impossible to define a general optimum value for MCG splitting, but rather a range of optimum values. Therefore, the optimum solution is to set a desired range of splits (which is predefined from 4 to 7). Then, for each MCG the value that minimizes the overlapping of the two last splits selected. This configuration has been tested and no unexpected results have been found, maximizing the motion information extraction while preventing from undesired effects.

We next discuss the resolution of the circular diagram. The circular diagram feature is the key to all other shape related features. Its angular and radial divisions are configurable, and the selected values affect the effectiveness of the shape related features. As in the case of movement splits, a poor resolution omits information while a large resolution can worsen the results due to noise or excessive information. Therefore, finding the optimum parameters is crucial. The classification performance of datasets 1 and 2 (discussed in Sect. 5.2.1) for different resolutions of the circular diagram is compared. Only MCGs used for the training have been used for recognition. In this way the results are only affected by how are the MCGs characterized.

Tables 5.6 and 5.7 show the performance of the classification performance for datasets 1 and 2. As motion related features are not used to classify dataset 2, the range of the configured MCG split is omitted. Analyzing these two tables, the optimum values for motion splits and circular diagram splits can be extracted. Increasing the radius to 5 proves to increase classification failures, while the angular division optimum is

<table>
<thead>
<tr>
<th>Circular diagram splits</th>
<th>Motion splits</th>
<th>4 - 6</th>
<th>5 - 7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class 1</td>
<td>Class 2</td>
<td>Class 3</td>
</tr>
<tr>
<td>4×8</td>
<td>96.5%</td>
<td>100%</td>
<td>99.5%</td>
</tr>
<tr>
<td>4×10</td>
<td>96.5%</td>
<td>99.5%</td>
<td>99.0%</td>
</tr>
<tr>
<td>5×10</td>
<td>64.5%</td>
<td>100%</td>
<td>65.0%</td>
</tr>
<tr>
<td>5×12</td>
<td>56.5%</td>
<td>99.5%</td>
<td>65.5%</td>
</tr>
</tbody>
</table>

Table 5.6: Ratios of MCG correctly classified for dataset 1.
Table 5.7: Ratios of MCG correctly classified for dataset 2.

<table>
<thead>
<tr>
<th>Circular diagram splits</th>
<th>Class 1</th>
<th>Class 2</th>
<th>Class 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4×8</td>
<td>88.0%</td>
<td>76.0%</td>
<td>74.5%</td>
</tr>
<tr>
<td>4×10</td>
<td>100%</td>
<td>99.5%</td>
<td>100%</td>
</tr>
<tr>
<td>4×12</td>
<td>91.0%</td>
<td>92.5%</td>
<td>98.5%</td>
</tr>
<tr>
<td>5×8</td>
<td>73.5%</td>
<td>56.5%</td>
<td>88.0%</td>
</tr>
<tr>
<td>5×10</td>
<td>96.0%</td>
<td>99.5%</td>
<td>99.0%</td>
</tr>
<tr>
<td>5×12</td>
<td>92.0%</td>
<td>96.0%</td>
<td>100%</td>
</tr>
</tbody>
</table>

around 8 or 10. Despite being 8 the optimum for these specific tests, its results are very similar to than of 10 angular splits. Therefore 10 is selected as the preferred option as it gives better resolution. However, if the classes to be classified are very different to these, the optimum circular diagram resolution should be reanalyzed to match with the requirements of the new classes.

5.3 Classification results

To start with let us recall that the practical application of the work presented here is the identification of stars observed by the Gaia space astrometric mission. The nominal duration of the mission is five years, during which Gaia will observe each star about 85 times. Since operations started just about one year ago, most of the objects have only a few scans for the moment being. This represents a problem for testing our algorithms on real data, as incomplete data is insufficient to characterize objects. However, extended objects yield many observations for each scan, so they are almost the only objects that can currently be used to test this project using real data. Extended objects classification is a challenge, as they are difficult to characterize in classes by shape or motion features. Even more difficult is to differentiate from one type to other. Another bypass solution to this lack of data is to use simulated data. Although it is not real data, it has the obvious advantage of being as abundant as needed. Therefore, this section is divided into simulated data and real data results.

5.3.1 Simulated data

To take advantage of the abundance of the simulated data, a large dataset of 5,000 MCGs has been designed, which should be consistent with the huge diversity of MCGs that will conform the final real data. To enhance the test, the simulation is done with high freedom in all MCG properties. The ranges for the most relevant simulated features are the following:
Chapter 5. Results and analysis

<table>
<thead>
<tr>
<th>Property</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of sources</td>
<td>1 to 7</td>
</tr>
<tr>
<td>Radius</td>
<td>100 - 2,500 mas</td>
</tr>
<tr>
<td>Observations per source</td>
<td>60 – 90</td>
</tr>
<tr>
<td>Distance between sources</td>
<td>50 – 250 mas</td>
</tr>
<tr>
<td>Proper motion mean</td>
<td>1,000 mas/yr</td>
</tr>
<tr>
<td>Proper motion variance</td>
<td>300 (mas/yr)^2</td>
</tr>
</tbody>
</table>

Regarding proper motion mean and variance, to have both HPM and non-HPM sources the simulator is configured with a random switch for the proper motion simulation. Therefore, some sources are simulated with proper motion within the configured range, while others are simulated without any proper motion. The resulting dataset has all types of MCG. Therefore, the training is done with five classes to maximize the classification capabilities. The classes trained are the combination of the recognition testing datasets 1 and 2, detailed on Sect. 5.2.1. The circular and HPM classes are expected to classify single sources that fit in its characteristics. Therefore these two classes are focused in shape and motion of a single source, while the other classes are focused on the shape and structure that all the sources form in the group. The two source class is expected to have mainly two source MCG but also others that have all the observations concentrated in two regions. The same is applied to the three sources class, where MCGs with a triangle shape are the main classification objective, independently of the number of sources. The five sources class is used to gather multiple source groups with different shapes. As the projection space has as dimension of the number of trained classes minus one, this classification results on a four-dimensional space. Therefore, illustrating the results with figures is meaningless. Fortunately, there are other ways to detail the results.

5.3.1.1 Visual inspection

We first inspected the circular MCG class. Inspection of the MCGs classified shows very good performance for this class. Only a few not expected MCGs are wrongly classified. Most of them, as those shown in Fig. 5.18, have all sources overlapped on the same region. We then continued with the HPM class. Inspection of the MCGs classified shows an excellent performance. All the classified MCG are HPM single sources, and none of them have been seen on other classes.

Next we discuss the two sources MCGs. As in the circular class, visual inspection shows very good performance. Some MCGs with more than two sources have been classified. However, they are expected as its shape is very similar to the one used for training.
Figure 5.18: Wrongly detected circular single source.

Figure 5.19: Wrongly detected two source MCG.

Figure 5.19 shows an example. Still, some two source MCGs have been skipped as their projection is too far from the center of the class. However, this could be due to the training class not being representative enough for all the possible combinations.

We continue describing the results for the three source MCGs. As expected, this class is composed mainly of MCGs with a triangular shape, independently of the number of sources. Some other shapes have been classified that are not exactly expected. However, it represents only about a 10% of the total and its shape is similar to a triangle. Figure 5.20 shows two of them.
5.3.1.2 Class homogeneity

Another way to analyze the results is by analyzing the homogeneity of the resulting classification. To do that, the Mahalanobis distance to the classified class for all the MCGs in the class is employed. It must be considered that this is a part of the classifying process.
Statistical analysis of a massive astronomical cross-matching process

<table>
<thead>
<tr>
<th>Class</th>
<th>Non limited distance</th>
<th>Limited distance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Standard deviation</td>
</tr>
<tr>
<td>Circular MCG</td>
<td>3.05</td>
<td>0.72</td>
</tr>
<tr>
<td>HPM MCG</td>
<td>3.15</td>
<td>0.66</td>
</tr>
<tr>
<td>Two source</td>
<td>1.62</td>
<td>0.66</td>
</tr>
<tr>
<td>Three sources</td>
<td>1.29</td>
<td>0.42</td>
</tr>
<tr>
<td>Five sources</td>
<td>2.06</td>
<td>1.18</td>
</tr>
</tbody>
</table>

Table 5.8: Results of the classification distributions.

rules. To improve the analysis two tests are done. One test with a maximum of two standard deviations distance and another with totally free maximum distance to the class. Table 5.8 show the results.

A well trained class should have all its projections focused near the center of the class, and other type of MCG should be projected far from it. Therefore, if classifications are not filtered by its distance to the nearest class, a large increase in the distance mean and variance is expected. On the other hand, a modest increase in the distance means that a significant number of MCG have its projections near the trained class but not near enough to be considered of this class. This can be produced by two reasons. There are MCGs similar to the trained class (Fig. 5.18 is an example). Therefore its projections fall near it. These objects are skipped by the maximum distance configured in the recognition system. If these objects are manually inspected and a they are similar between them, they can be added to the training as a new class so the LDA algorithm tries to isolate them from other classes. Another possibility is that the training dataset does not represent the entire class that is going to be classified. Therefore, MCGs that are not similar to the ones used for the training are scattered around.

Analyzing these results some conclusions can be extracted. As can be seen, for the circular and HPM classes, with no limit on the maximum distance, the mean and variance increase by similar ratios. This is because of similar MCGs fall far from the class. Therefore, these cases can be easily discarded. For the two and three sources the classes were trained not because of its number of sources but because they have a typical shape. Other combinations of sources that have the same shape or very similar are intended to be classified. Manually selecting all the possible cases with this shape is far more cumbersome that only a few. Therefore, the projections of these MCGs are more dispersed around the class. Nevertheless, this is expected for these two classes. Also, considering the extremely small increase in the distance mean and variance, MCG projections can be found almost only in the vicinity. Therefore, these classes are well isolated from other cases. The most remarkable feature of the results obtained for the five source class is the drastic increase in the standard deviation of the distance. This is due to the fact that projections that fall around this class are extremely far and only
Chapter 5. Results and analysis

### Results and analysis

69

<table>
<thead>
<tr>
<th>Property</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radius</td>
<td>100 – 1,000 mas</td>
</tr>
<tr>
<td>Observations per source</td>
<td>50 – 80</td>
</tr>
<tr>
<td>Proper motion mean</td>
<td>0 mas/yr</td>
</tr>
<tr>
<td>Proper motion variance</td>
<td>100 (mas/yr)$^2$</td>
</tr>
</tbody>
</table>

**Table 5.9: One source MCGs simulated parameters.**

<table>
<thead>
<tr>
<th>Property</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radius</td>
<td>100 – 1000 mas</td>
</tr>
<tr>
<td>Observations per source</td>
<td>50 – 80</td>
</tr>
<tr>
<td>Distance between sources</td>
<td>50 – 150 mas</td>
</tr>
<tr>
<td>Proper motion mean</td>
<td>0 mas/yr</td>
</tr>
<tr>
<td>Proper motion variance</td>
<td>100 (mas/yr)$^2$</td>
</tr>
</tbody>
</table>

**Table 5.10: Two source MCGs simulated parameters.**

very few. This class is designed to pick up all multi-source MCGs with strange shapes that do not fit in other classes. Therefore, as its definition is as broad as its training and as MCGs with up to 7 sources can form really unusual combinations, this class is trained to have many different MCGs. Therefore, the objects that fall near the class boundary will be very different to it, as any small similarity would largely decrease the projection distance to this class. This explains that MCGs entering the classification when the maximum distance is unlimited are very distant.

5.3.2 Real data

In order to get the data classified, first a manual inspection of the available classes is done. As commented previously, due to the lack of observations at this stage of the mission, most of MCGs are extended objects, such as galaxies and nebulae. These classes are difficult to classify, as their shapes occupy a wide range of options. Therefore, is expected that some of them will avoided during the classification, because the system will be unable to clearly link them to a class. Additionally, some other MCGs composed of one or two sources are found with sufficient observations to be analyzed. The classification is done with a minimum of 45 observations per MCG to enter the classification system. The training is done with the following classes.

The first of these classes consists of one-source MCGs. Due to the lack and difficulty to manually find these MCGs in the real data available so far, the training is done with simulated MCGs. The relevant simulated parameters are shown in Table 5.9. Quite naturally, the second one corresponds to two-sources MCGs. As it occurs with the one-source MCGs due to the lack of available data for this class, the training is also
done with simulated data. The relevant simulated parameters are shown in Table 5.10.

We additionally studied galaxies. To this end we manually selected galaxies from the available real data. Figure 5.21 shows an example of a galaxy used for the training. Finally, we also manually selected nebulae from the available real data. Figure 5.22 shows an example of a nebula used for the training. The challenge of these two last tests is to correctly detect and classify extended objects. The one- and two-source classes are only to allow the LDA algorithm to isolate these MCGs projections from the galaxies and nebulae so the classification can be done reliably. Also, the more remarkable features that can differentiate galaxies from nebulae are unnoticeable. The resolution of the circular diagram for this classification has been configured in such a way that it can measure the small empty space that galaxies use to have in the center with the radial mean feature. Also, nebulae use to concentrate observations along remarkable structures. Thus, with enough resolution the radial variance feature should detect the difference.

To summarize, the statistics used are MCG flattening, magnitude variance, radial linearity, radial mean, observations per scan and cluster counter. The resolution adopted for the circular diagram feature is 7 radial divisions and 10 angular divisions. This configuration has been found evaluating the classification performance of the training samples. However, while it is the best configuration found to characterize extended objects, it is not the best one for other MCGs as the two-source MCG. Nevertheless, the main objective is to classify extended objects, so it is prioritized over other classifications.

As galaxies and nebulae can be very different from each other within the same class, the training is done with objects that are similar. This is especially true for galaxies, as nebulae are even more different from each other. If the training is done with very different galaxies, the resulting class distribution could be so large that includes other classes. To avoid this, the training is done giving priority to the homogeneity of the classes, at the cost of other not similar MCGs classifications being avoided. In other words, the validity of the classification is prioritized at the cost of the number of classified MCGs.

Table 5.11 shows the number of MCGs classified for each class. Before detailing the results, an issue with the real data obtained must be explained. It has been found that Gaia has an onboard issue that leads to spurious (false) detections. These spurious detections use to be quite aligned and usually lie around real stars, specially bright ones. The resulting MCG has features that resemble that of the galaxies. The top panel of Fig. 5.23 shows an example of these cases. Also, sometimes these are far enough from the source to look like isolated. In these cases the spurious detections use to be a long line of observations. The bottom panel of Fig. 5.23 shows an example. However, this
Figure 5.21: Trained galaxy.

<table>
<thead>
<tr>
<th>Class</th>
<th>Number of classified MCGs</th>
</tr>
</thead>
<tbody>
<tr>
<td>One source</td>
<td>125</td>
</tr>
<tr>
<td>Two sources</td>
<td>96</td>
</tr>
<tr>
<td>Galaxies</td>
<td>901</td>
</tr>
<tr>
<td>Nebulae</td>
<td>26</td>
</tr>
<tr>
<td>Avoided</td>
<td>7699</td>
</tr>
</tbody>
</table>

Table 5.11: Classification results on real data.

As previously explained, this classification is expected to be difficult. Therefore, avoiding classification errors has been prioritized, rather than trying to classify the maximum number of MCGs. This has been done configuring strict decision parameters. Consequently, many MCGs are avoided. However, this is done in such a way that the resulting classification is correct and homogeneous. We conducted a manual inspection of the outcome to evaluate the results. For one-source classes the results show a very good classification performance. Most of the MCGs have a number of observations near the 45 configured limit, which was expected. Manual inspection shows that all the MCGs analyzed have one observation per scan, and uniformly distributed observations around the source. Therefore, the classification of this group is correct. For two-sources classes we find that most of the MCGs in the class are clearly two source MCGs. However,
other MCGs with elongated shapes have been classified. Nevertheless, they are about 10% of the total, and the ratio of observations per scan is always small, indicating that are not extended objects. However, the classification of this class is quite good. Regarding galaxies, manual inspection of the results shows that the classification is very good. Galaxies are the most common objects within the actual data with sufficient observations, therefore is the larger group. Some of the classified MCGs have few scans, which makes difficult to manually identify the shape. However, they are clearly extended objects, and are more alike to a galaxy than to a nebulae. Some spurious like the shown in the top panel of Fig. 5.23 have been also classified in this group. Although they are an important fraction of the group (about 25%), this is the expected behavior, which is due to the similarities, and little can be done to avoid them. Finally, we discuss the results for nebulae. All the 26 MCGs classified in this group are nebulae, and all of them have very different shape. Note that this is the more difficult MCG to classify, since each nebulae is completely different from the rest. However, as the training done is homogeneous enough, nebulae with similar features have been detected. Therefore, even considering the low number of classified objects, we consider that this classification is a great success of our method. Figure 5.24 shows two examples of detected nebulae.
Figure 5.23: Some examples of spurious detections in real data. The top panel shows spurious detections around a source, while the bottom panel displays an isolated spurious detection.

spurious detections. As the spurious detections surrounding a source show many similarities, the spurious being classified are the ones like the shown in the bottom panel of Fig. 5.23. The approach followed consists of computing only the radial linearity and radial mean features with a high angular resolution of 16 divisions. Therefore, these spurious are expected to be the only objects with a radial linearity of 1.0, and observations uniformly distributed over all radius. With only 3 cases used for the training, the system has been able to find 11 more, which has been used to increase the training dataset. With the increased training dataset, the system classified 17 MCGs, of which three are not real spurious. Therefore, a total of 14 spurious were found. It must be
said that these spurious are not as abundant as the others, but they are more easily detected, as the others can form any type of structure of fake detections around stars.

After the first successful results of the classification over the selected real data, another classification has been performed over all the accumulated data available from Gaia so far, which represents about 400 Gigabytes of data. It is unfiltered and contains many spurious detections, specially of the type shown in the top panel of Fig. 5.23. Table 5.12 shows the number of classified MCG per class.
Manual inspection of the one- and two-sources classes shows similar results to those for the selected data test. However, galaxies and nebulae classifications are plenty of spurious. What initially can be seen as a failure is, in fact, another achievement, as the spurious are classified as well. Similarities from the spurious and galaxies groups have been already discussed. However, isolated aligned spurious also have similarities with nebulae. It can be seen that both have observations which uniformly distributed radially, and a high radial variance (for a certain number of angular divisions). Whatever the case is, for this configuration these features resemble each other. Despite these spurious can be easily detected with its specific configuration, they are rather similar to nebulae and, thus, the system is also useful for them. Therefore, we have found an effective way to detect and classify spurious detections, which are an important issue derived from the Gaia operation and cannot be avoided.

### 5.4 Computational load

Gaia is producing extremely large amounts of data, which is being accumulated for processing and analysis. Rather than a one-time step, this process is cyclic. Each iteration is based on the previous one to improve the quality of the output. Therefore, the computational load of the programs is a very important issue for the success of the mission. The computational load of this project has been evaluated using the data obtained after the last classification over all the accumulated data. This is a reasonable strategy, since the computing times necessary to initialize and train the algorithms are negligible. The time required to processed the 400 Gigabytes of data mentioned is 6 hours and 28 minutes, leading to an outstanding throughput of 17.6 MB/s — all this in an otherwise typical desktop computer. This is due mostly to the excellent performance of the circular diagram approach, which has demonstrated a very good performance in characterizing MCGs shape and largely decreases the amount of data to be processed.
Chapter 6

Conclusions

6.1 Conclusions

In this work a classification system for MCGs has been developed. The aim has been to delivery a system that helps developers to better understand the properties of the different types of MCGs and classify them into similar sets. To do this, a set of statistics or features that describe the properties of the groups has been designed and implemented. The performance of the proposed algorithm has been tested using simulated and operational data.

Working in a complex project like Gaia/DPAC has been a challenge. Considering each step in the system as an independent part has been key to test and find the optimal combination and configuration of the algorithms. The ability to simulate data for testing purposes has been critical as well. Being able to simulate data to test specific parts of the algorithm has allowed to design a better and more versatile system. It has allowed several strategies for each stage of the system to be tested and compared to select the best ones. Another conclusion of this work is that it has become clear that there is no bad classification algorithm, but rather there are bad or good datasets for each algorithm. Through the configuration of the computed statistics, the data which is input to the classification stage can be customized to take the maximum advantage of the algorithm.

The system implemented in this work has demonstrated an excellent performance with simulated data. Moreover, it has shown a very good performance also with real data. Even with the limitations of the data in this early stage of the mission, the system has been able to classify a significant fraction of the MCG into consistent categories. Additionally, objects that are not known to the system have been separated into a special category. These can be further analyzed manually or classified with other parameter in
subsequent executions of the classification system. It is also worth mentioning that a by-product of this work is the ability to classify with an excellent performance MCGs resulting from spurious on-board detections. This feature has proven to be extremely useful to developers and scientists working on Gaia as it will allow them to easily classify the remaining spurious detections on the data and therefore analyze the performance of their spurious detection algorithms. This is of particular importance, since Gaia produces extremely large amounts of data that has to be analyzed by the scientific members of DPAC before being released to the public. Hence, an efficient classification algorithm that can filter this data will be very useful and simplify the analysis.

Finally, due to the very large amounts of data (and records) involved in the Gaia mission, the performance of the algorithm is key. The approach taken for the feature computation and the projection of the features for the classification has demonstrated an excellent computational behaviour. The performance of the system has allowed us to run the classification over more than a year of Gaia data in a desktop computer in just a couple of hours.

### 6.2 Future work

During the execution of this work multiple interesting research lines have been uncovered. Unfortunately, these new research lines are out of the scope of the current work. However, we present here the one we judge to be the most important one, as it might be part of future developments. We have already indicated that our classification system has shown a good performance both in terms of reliability and computing load. However, as any statistical classification system, its performance is strongly dependant on the training stage. Thus, it would be interesting to explore other classification algorithms to see how they perform. Some of the newest classification algorithms can operate without supervision, meaning that given a dataset the algorithm looks for patterns and clusters in the data for itself. Rather than training some desired classes, it could find all the different classes that can be isolated, without human interaction. In particular, it would be specially interesting to employ Self-Organizing maps (SOM) to classify the data, and compare its results with those presented here.
Bibliography


