STRUCTURAL ANALYSIS OF ASTRONOMICAL IMAGES: A TWO STEP METHOD BASED ON GAUSSIAN MIXTURE REPRESENTATIONS

VILLANUEVA AHUMADA, MARTÍN ANDRÉS

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE

Structural Analysis of Astronomical Images: A Two-Step Method Based on Gaussian Mixture Representations

Martín Andrés Villanueva Ahumada

Undergraduate thesis presented in partial fulfillment of the requirements for the degree of
INGENIERO CIVIL EN INFORMÁTICA

Advisor	:	Claudio Torres López, Ph.D.
Co-Advisor	:	Mauricio Araya López, Ph.D.
Correferent	:	Marcelo Mendoza Rocha, Ph.D.

Acknowlegdments

To my mother, for bringing me to the world and for her unconditional love.

To my father, for the love, support and trust he gives me.

To my brother Tatán, for believing in me more than anyone.

Resumen

La identificación y cuantificación automática de componentes y fuentes extendidas, es un desafío clave para el análisis de datos astronómicos. El aumento de volumen, resolución y sensibilidad de las observaciones, requiere de nuevos algoritmos para la identificación de fuentes. Adicionalmente, la naturaleza jerárquica y acoplada de estructuras complejas, como las nubes moleculares, requieren que estos algoritmos sean capaces de recuperar las relaciones jerárquicas entre las fuentes. En este trabajo se propone un nuevo enfoque para la identificación de fuentes en datos astronómicos. En este, primero se representan los datos como una mezcla de Gaussianas, y luego se realiza un análisis automático basado en la reducción de componentes de la mezcla. Se identifican las fuentes y sus relaciones jerárquicas directamente sobre tal representación. Esto es diferente de otros algoritmos de identificación de fuentes por dos razones: 1) Aquí se realiza la identificación sobre una representación continua, en vez de ocupar la representación discreta de pixeles, tomando ventaja de las propiedades de la continuidad. 2) Se construye un árbol jerárquico para obtener las relaciones entre los componentes. Esto nos permite seleccionar representaciones con diferente número de componentes identificados, a un bajo costo computacional. Se evalúa la propuesta cuantitativamente y cualitativamente usando datos desde el archivo de ALMA Science verification, y además sobre datos generados sintéticamente. También se comparan los resultados con algoritmos de identificación de fuentes del estado del arte. Los experimentos y comparaciones realizados muestran que la propuesta es un modo efectivo de inspeccionar y representar la estructura jerárquica de las fuentes. El método fue capaz de identificar y representar correctamente las emisiones acopladas de fuentes sobrepuestas.

Palabras clave: Identificación de fuentes, Representación Gaussiana, Reducción de mezclas de Gaussianas, Análisis Estructural.

Abstract

The automatic identification and quantification of components in extended sources is a key challenge for astronomical data analysis. The increasing volume, resolution and sensitivity of observations, requires novel algorithms to identify sources. Additionally, the intrinsically hierarchical and blended nature of complex structures, as molecular clouds, requires these algorithms to be able to retrieve the hierarchical relationships between sources. We propose a novel approach for the identification of sources in astronomical data. It first represents the data as a mixture of Gaussians and then performs automatic analysis based on reduction of the mixture components. We identify the sources and its hierarchical relations directly over the mixture representation. This is different from other source identification algorithms for two reasons. 1) We perform the identification over a continuous representation instead of using discrete pixel representation, taking advantage of the continuity properties. 2) We build a hierarchical tree to obtain the relationships among the components. This allows us selecting representations with a different numbers of identified components with no extra significant computational cost. We asses our proposal quantitatively and qualitatively using data from ALMA Science verification archive, and also with synthetic generated data. We also compare the results with state-of-the-art source-identification algorithms. The experiments and comparisons show that our approach is an effective way to inspect and represent the hierarchical structure of sources. Our method was able to accurately identify and represent the blended emission of overlapped sources.

Key words: Source Identification, Gaussian Representation, Gaussian Mixture Reduction, Structural Analysis

Glossary

- **GMR**: Gaussian Mixture Reduction.
- **PDF**: Probability Density Function.
- **IMF**: Initial Mass Function.
- **FITS**: Flexible Image Transport System.
- **CAA**: Clump Assignment Array.
- **LM**: Levenberg-Marquardt algorithm.
- **KLd**: Kullback-Liebler divergence.
- **RMS**: Root Mean Square value.
- $f(\mathbf{x})$: Function representation of the original image.
- $f_0(\mathbf{x})$: Function representation of the observed image.
- ϵ: Background additive noise.
- $u(\mathbf{x})$: Function in the space of possible solutions.
- u_0: Base level
- Ω: Domain of the input data; $\mathbf{x} \in \Omega$.
- $\partial\Omega$: Boundary of the domain Ω.
- D: Dimensionality of the input data (2 for images, 3 for cubes).
- $\boldsymbol{\mu}_i$: Center of the i-th Gaussian.
- $\tilde{\boldsymbol{\mu}}_i$: Reference center of the i-th Gaussian.
- $\boldsymbol{\delta}_i$: Neighborhood ratio of the i-th Gaussian.
- Σ_i: Covariance matrix of the i-th Gaussian.
- $\sigma_{\min}$: Minimal Gaussian width (standard deviation).
- $\sigma_{\max}$: Maximum Gaussian width (standard deviation).
- BSize: Beam Size.
- ARes: Angular resolution.
- $\boldsymbol{\eta}$: Parameters vector, it parameterizes the space of solutions.

- Ψ: Penalty function for the domain restrictions.
- α: Parameter that controls the maximum amount of penalization in domain restrictions.
- λ: Parameter that controls the speed of penalty in domain restrictions.
- $\mathcal{P}$: Set of center points.
- $\hat{\mathcal{P}}$: Set of reference center points.
- $\mathcal{Q}$: Set of collocation points.
- $\mathcal{B}$: Set of boundary points.
- $d_{\mathrm{KL}}(i, j)$: KL-divergence upper bound between components i-th and i-th of a mixture.
- $\mathcal{I}$: Set with indexes of the usable pixels.
- $\mathbb{R}$: Set of real numbers.

Contents

List of Figures

1 | Introduction

In this work we address the problem of identifying and representing extended emission sources in astronomical observations. In particular, we focus on images and spectroscopic cubes with relatively diffuse and overlapped structures within the interstellar medium (ISM), such as molecular clouds, HII regions or other nebulae. The study of these sources, also known as clumps, is very important to understand star and galaxy formation. Compact components within the ISM (or *clumps*), are the precursors of core structures that will become more classical astronomical sources such as disks, stars, or planets. This allows us to study their very early stages of formation (Kennicutt Jr and Evans, 2012). Unfortunately, the detection of these components becomes a challenging problem when the volume, resolution and sensitivity of observations increases. Moreover, the analyzed regions can be highly non-homogeneous in their density, and may contain a large number of components in very overlapped and hierarchical configurations (Alves et al., 2007).

Several procedures have been proposed in the last years for tackling the problem of automatically detecting clumps (Berry et al., 2007). In a first approach, we divide them into those that work using discrete representations and those that uses a continuous representation. Since images and cubes are available in discrete pixel-based formats (e.g., FITS), the most common approach is to separate clumps directly on the pixels domain. Most of these algorithms associate each pixel to a single clump, segmenting the image into disjoint sets of pixels (Williams et al., 1994; Bertin and Arnouts, 1996; Berry, 2015). This neglects the blended nature of the components of a cloud, which might be a combination of the flux contribution of multiple components. An alternative to this, is to represent the data as a combination of functions in a continuous domain, as proposed by the *gaussclumps* algorithm (Stutzki and Guesten, 1990). This allows representing overlapping clumps using only a few parameters per each of these clumps. However, this early approach has many limitations and strong assumptions, such as strict *Gaussianity* of clumps.

An important problem with automatic identification assessment is the lack of *ground truth* knowledge about the distribution of sources present in the emission. This unavoidable uncertainty makes difficult the proper assessment of source identification algorithms. An alternative approach to address this difficulty, is to handle the identified sources in a hierarchical way, without a fixed number of components. This requires delivering a tree-like data structure as output. The gradual decomposition of components and sources casn be done then by the user, depending on the science case the user is studying. Dendrograms (Rosolowsky et al., 2008) is the most known approach of hierarchical representations, but it has the limitation of working only with discrete representations.

Our proposal consist on mapping the data from the discrete pixels representation to a continuous one before identification. Specifically, we use a Gaussian mixture to represent the data without assuming any particular shape of the clumps: there is no one-to-one correspondence between clumps and Gaussian components of the mixture. Then, we perform the source identification directly over the continuous representation, through the use of the mathematical properties of the mixtures of Gaussian. In order to provide an accurate representation, we minimize a functional, following a variational approach (Logan, 2013), similar to what is done for image denoising or image restoration (Wang et al., 2014). To make the solution computationally feasible, we make some assumptions over the model, which allows us to get a successful continuous representation. The process of identification of clumps over the Gaussian mixture representation is done with Gaussian Mixture Reduction (GMR) algorithms (Crouse et al., 2011). We generate a hierarchical binary tree similar in purpose to Dendrograms (Rosolowsky et al., 2008), but with a different structure. Instead, we decompose hierarchically each of the sources, generating representations with a variable number of sources identified. This can be done automatic or in a user supervised fashion. Unlike Dendrograms, this structural analysis remains in the continuous space, i.e., the source are decomposed such that the constituents components, are still represented as Gaussian mixtures.

Our two main contributions to the source identification problem are: 1) the construction of a continuous representation of the data with a mixture of Gaussians, which reduces the noise and avoid adding extra flux, and 2) propose a novel mechanism to perform the structural analysis, describing the hierarchical relations of the sources with continuous representations, allowing to capture the blended nature of the emissions.

This document is organized as follows, in Chapter 2 we introduce and explain the problem to be addressed, in Chapter 3 we describe and discuss the state-of-the-art methods used for the astronomical community for source representation and description. Chapter 4 introduces the continuous representation model corresponding to the first step of our proposal, while Chapter 5 presents the identification over the continuous representation, corresponding to the second step. Chapter 6 presents experiments to assess quantitatively and qualitatively the capabilities of our method. Finally Chapter 7 presents the conclusions and future work, respectively.

2 | Problem Description

To properly understand the problem addressed in this work, we need to introduce the kind of astronomical data we work with and its format. We are interested in observations of interstellar molecular clouds, which are composed by cold dust and gas. They are important because they are the only regions of the space where the formation of some more compact astronomical objects take place. Examples of these objects are stars, disks, planets, galaxies, among others.

These observations come in two formats: As 2-dimensional images in the *continuum*, or as 3-dimensional spectroscopic cubes. The first has two celestial coordinates, corresponding to the *Right Ascension* (RA) and the *Declination* (DEC). The second adds a third coordinate corresponding the the observed frequency. The images captures a region of the space over a wide range of the electromagnetic spectrum (in the continuum), where all these frequencies are collapsed in a single bi-dimensional array. The cubes captures a region over a specific frequency band, producing a frequency spectrum for each pixel. An schematic of an spectroscopic cube is shown in Figure 2.1.

Our goal is to identify and characterize the sources in the molecular clouds, also known as *clumps*. A clump can be generally understood as a region of high density within a larger area of lower density, where this density is measured over some physical scalar property (the most common are flux and intensity). These molecular clouds are usually non homogeneous in their density, and are composed of cores of high density gas matter contained and separated by low density areas (Kennicutt Jr and Evans, 2012). Each observation of such clouds, may contain multiple source cores and clumps in complex configurations both in spatial and spectral domains.

Manual identification of clumps becomes impractical for observations with high resolution axes, and as the number of observations increases with each new instrument, the automation of this process becomes critical. Moreover, as the image becomes more crowded with clumps, problems like blended emissions arise, where multiple dense cores converge in a nearby area. In these cases,

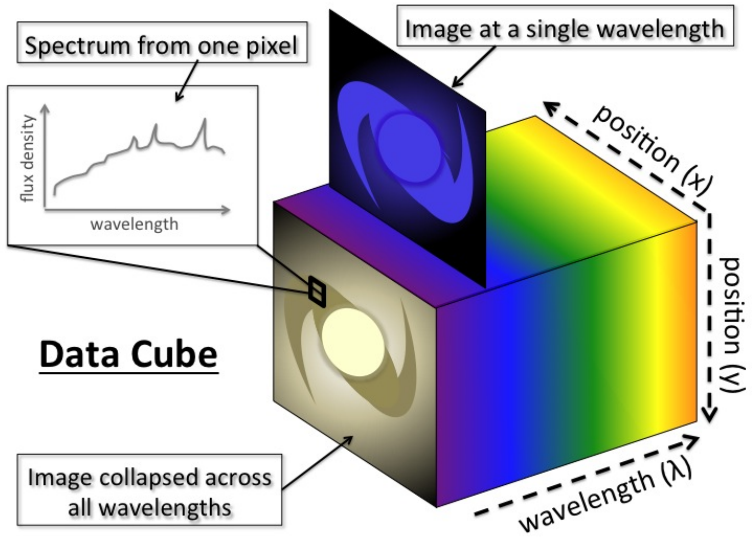


Figure 2.1: Schematic diagram of an spectroscopic data cube.
(Image obtained from `http://astro.dur.ac.uk/~cpnc25/research.html`)

subjective biases become more evident as each astronomer might perceive the data differently.

However, identifying clumps is not enough: each dense clump is often contained on lower density cores, constituting a hierarchical structure (see the sources contained within other sources in Figure 2.2). Consequently, the challenge also includes determining the hierarchical relationships between clumps, a task also called structural analysis. This information is crucial to understand processes such as star formation (Alves et al., 2007).

One of the main difficulties of this problem is the lack of *ground-truth* knowledge about the real distribution of sources in the observations. This means that it is not possible to know a priori the number of sources and its flux distribution. This makes the validation and assessment of the clump identification algorithms a challenging task.

2.1 Goals

The main goal of this work is to propose, develop and assess a novel method for the identification and hierarchical characterization of astronomical sources. All this based on a continuous representation of the data, through Gaussian mixture representations. This method is intended to be a tool to help astronomers in the analysis of astronomical data. In particular, we want to contribute in the task of structural and hierarchical analysis of sources.

Figure 2.2: Extinction map of the Perseus star-formation region.
(Image obtained from `https://dataverse.harvard.edu/dataset.xhtml?persistentId=hdl:10904/10080`)

Below we present a list with the specific goals of this work:

1. Create a theoretical model for the continuous representation of astronomical data, based on mixture of Gaussians, calculus of variations and least squares minimization.

2. Propose a source identification mechanism over the mixture representation, taking advantage of its theoretical properties.

3. Create a hierarchical tree data structure that allows us to study the structure of the sources, in a convenient way.

4. Develop a Python software package that implements our proposal, and integrate it into the ACALIB library. [†].

5. Quantitatively evaluate the accuracy of the obtained continuous representation, and its robustness against different parameters configuration.

6. Assess the robustness of our proposal compared to some of the state-of-the-art source identification algorithms.

[†]Advanced Computing for Astronomy Library

7. Show qualitatively and illustratively how our method works and the benefits it provides for the structural analysis.

3 | Related Work

Several procedures have been proposed in the last years for tackling the problem of automatically detecting sources and clumps in astronomical data. In this chapter we provide a non-extensive but significant review of them, to put our method into context.

3.1 Discrete Representations

Since images and cubes are available in discrete pixel-based formats (e.g., FITS), the most common approach is to separate clumps directly on the pixels domain.

A successful algorithm of the class that uses discrete representations is *clumpfind* (Williams et al., 1994). It was motivated by how the human eye identify structures on images. how the eye decomposes the maps into clumps by looking at the contour levels in a gradually decreasing way. This algorithm works by contouring the data at level sets, multiples of the RMS of noise. It starts from the highest level, where isolated cores are identified and each of these are considered as a new clump. Then this step is repeated by gradually decreasing the level set, identifying isolated structures and connecting them with clumps identified at higher levels, or defining them as new clumps otherwise. When a clump identified at a given level is connected to more than one clump identified at a previous level, then this structure correspond to a blended emission, and is divided (see Figure 3.1). To do that a *friends-of-friends* strategy is used, assigning each pixel of the identified structure to one of the adjoining clumps (the one *with more friend* pixels). It produces as output a discrete pixel representation called *Clump Assignment Array* (CAA), where each pixel is assigned exclusively to one clump identifier. A key characteristic of *clumpfind* is that no a priori clump profile is assumed, and then the identified clumps can have any shape.

The *SExtractor* (Souce Extractor) (Bertin and Arnouts, 1996) is probably the best-known and

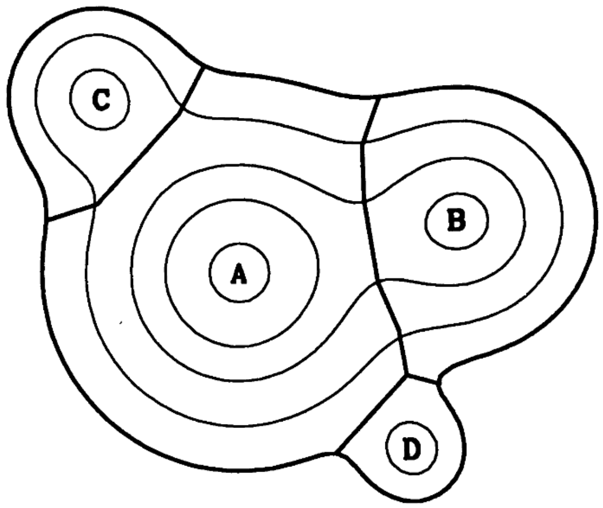


Figure 3.1: Example output that the *clumpfind* algorithm produces.
(Image obtained from Williams et al. (1994)'s paper)

mature algorithm that identify sources within a discrete representation. The main steps of this algorithm are: measure the background and RMS noise, subtract the background from the image, convolve the image with specific profiles, find isolated objects with thresholding, deblend emissions of overlapped objects, measure the positions and sizes of the objects using intensity moments, and finally catalog them. Thanks to the deblending of merged objects stage, this algorithm handles the problem of blended emissions. For each pixel which has flux contributions from more than one source, a probability of belonging to each of these sources is estimated, and then the pixel is assigned to the source with greatest probability. Anyway, the observed pixel flux value cannot be shared between two or more sources. This method is designed to work with optical and near-IR images in extra-galactic astronomy.

A more recent algorithm is *fellwalker* (Berry, 2015). This is a gradient-tracing scheme. The gradient is used in order to reach a local emission peak, which defines a new clump. The *fellwalker* algorithm, like *clumpfind*, divide the data in disjoint regions, each associated with a single significant peak, producing a CAA. This algorithm works by computing ascent paths (with the highest gradient) for each single pixel, as shown in Figure 3.2. These paths have two alternatives: 1) reach a local peak and therefore finding a new clump. In that case a new search it is performed on a wider neighborhood, verifying that it is not a noise spike. 2) reach another ascent path computed previously, and assign the pixels on that path to the corresponding clump identifier. At the end of the algorithm, detected clumps with low dip between them, are merged as a single clump.

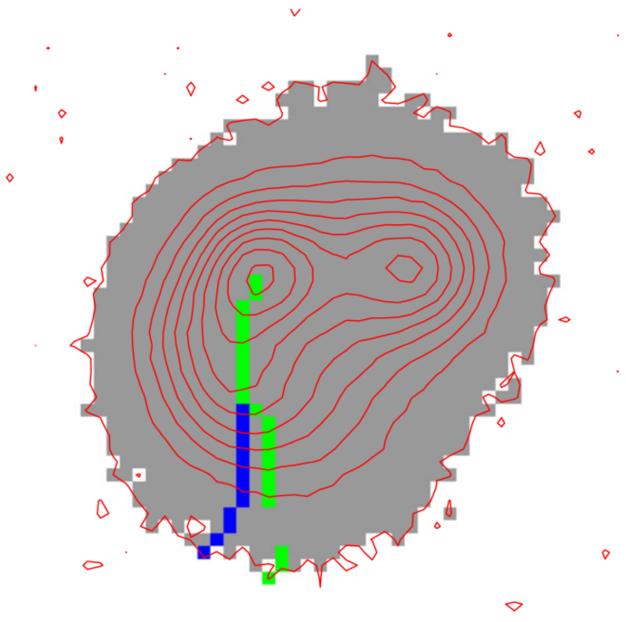


Figure 3.2: Two ascent path computed by the *fellwalker* algorithm in a 2D image.
(Image obtained from Berry (2015)'s paper)

A compelling analysis and assessment of the clumping algorithms implemented in the CUPID software package Berry et al. (2007) (*gaussclumps*, *clumpfind*, *reinhold*, *fellwalker*) can be found in (Watson, 2010). In particular they tested the robustness of the results of the algorithms, and its sensitivity to different configuration of parameters.

3.2 Continuous Representations

The algorithms presented in the previous section associate each pixel to a single clump, dividing the image into disjoint sets of pixels (Williams et al., 1994; Bertin and Arnouts, 1996; Berry, 2015). This neglects the blended nature of the components of a cloud, which might be a combination of the flux contribution of multiple sources.

An alternative to pixel level analysis, is to represent the data as a combination of functions in a continuous domain, as proposed by the *gaussclumps* algorithm (Stutzki and Guesten, 1990). This algorithm is the only well-known clump identification algorithm which works with continuous representations. The main idea of this, is to adjust Gaussian profiles that best fits each of the emission peaks. This algorithm iteratively searches for the current emission peak, and fits a Gaussian function (searching for the parameters of that Gaussian) constrained to keep the position and amplitude of the fitted shape close to the image maximum. This is done through a least squares minimization. Then the fitted Gaussian is subtracted from the signal, and the same step is repeated over the residual cube. The iteration continues until the flux of all subtracted clumps is equal to the integrated flux of the original image, or there are no significant peaks left. As result we get a set a Gaussians with different weights, locations and orientations. The sum of all these Gaussian components plus the background noise, allow to reconstruct the signal. In this algorithm each Gaussian is considered as a single clump, and because Gaussians could overlap between them, then it is possible that pixels *“are assigned"* to more than one clump. This allows representing overlapping components using only a few parameters per component. However, this early approach has many limitations such as several free parameters to set the stopping criteria, and strong assumptions, such as strict *Gaussianity* of clumps.

Recently, (Araya et al., 2017) also proposed a continuous representation of the data by using *Gaussian Homogeneous Representations* (GHR) of the spectroscopic cubes. GHR represents the data as a mixture of homogeneous and compactly supported Gaussians, which they called *bubbles*, because of the bounded flux of each of them. This representation is reached through an iterative subtraction method, that they proved to be computationally bounded. They also proved statistical independence between the *bubbles*, so the data analysis techniques which require independent and identically distributed sampling are exploited over this representation.

3.3 Hierarchical Representations

Astronomical images are frequently crowded of sources at very different spatial scales and wavelengths. In this cases, extract the sources with a plain source identification algorithm might not be enough. A better approach is the *structural analysis* of them through hierarchical representations, which produce a characterization of the sources hierarchical structure.

Alves et al. (2007) studied the origin of stellar *Initial Mass Function* (IMF) of interstellar molecular clouds, through the use of multi-resolution analysis with *Stationary Wavelet Transforms* (SWT). The clump identification procedure first maps the corresponding image to the wavelet space at different scales, then it identifies isolated structures at each scale, and finally connect them to neighboring structures in the successive scales. With this information they build a 3D distribution of significant peaks, and find the independent trees corresponding to one core and its corresponding hierarchical details.

A different discrete description of clumps is done with dendrograms (Rosolowsky et al., 2008). This method is conceptually different from local segmentation algorithms like *clumpfind*, since it aims to track the hierarchical structure over a range of scales. The dendrogram of a data cube is an abstraction of the changing topology of the isosurfaces (surfaces with the same intensity value) as a function of contour level (see Figure 3.3). Points in the dendrogram structure correspond to specific volumes in data cubes, defined by their bounding isosurfaces. The authors present this technique as a statistical and hierarchical representation of the molecular ISM.

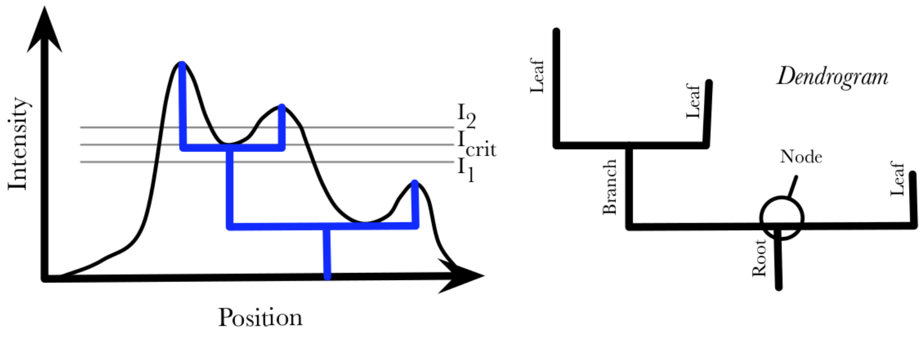


Figure 3.3: Schematic of Dendrograms mechanism. Thresholding at level I_1 produces a single identified object, while that thresholding at I_2 allows to decompose it, identifying two different objects.
(Image obtained from Rosolowsky et al. (2008)'s paper)

Men'Shchikov et al. (2012)'s *getsources* algorithm is a multi-scale and multi-wavelength source extraction method. It analyzes fine spatial decompositions of original images across a wide range of scales and across all wavebands. It cleans those single-scale images of noise and background, and constructs wavelength-independent single-scale detection images that preserve information in both spatial and wavelength dimensions. Sources are detected in the combined detection images by following the evolution of their segmentation masks across all spatial scales. Measurements of the source properties are done in the original background-subtracted images at each wavelength. The background is estimated by interpolation under the source footprints and overlapping sources are deblended in an iterative procedure.

Gregorio et al. (2014)'s approach follows a very similar methodology as Alves et al. (2007). They perform SWT over images to get a multi-resolution view of the data. However they were not satisfied with the thresholding step. To solve this, they proposed to use the clump identification algorithms from CUPID (Berry et al., 2007), to segment the data at the different levels on the Wavelet space. Then they built a hierarchical structure which captures the hierarchical relations between the sources in neighbor levels. Villanueva and Araya (2017)'s work extended this idea to work with spectroscopic cubes. They used 3D discrete wavelet transform for the multi-resolution analysis of the cubes. For the identification and construction of the hierarchical tree, they used the *fellwalker* algorithm.

It is important to mention that despite the hierarchical focus of the algorithms presented above, they remain in the discrete representation level. In others words, no matter the output these algorithms produce, they still associates each pixel to a single source.

4 | First Step: Continuous Gaussian Mixture Representation

We propose a two-step method composed of the following stages: 1) First we find a *continuous approximation* of the data and then, 2) We perform a *source identification* over the continuous representation. The idea is to compute an accurate continuous representation of the n-dimensional astronomical image, and then use this representation to extract the clumpy structure of sources in the data.

The *continuous approximation* step is closely related to Stutzki and Guesten (1990)'s *gauss-clumps*, but we propose to fit a Gaussian mixture in a global and coupled way, as we describe in this section. The *source identification* is performed through Gaussian Mixture Reduction (GMR) algorithms, particularly the one proposed by Runnalls (2007), which is later explained in Chapter 5.1.

4.1 Data Pre-processing

We work with astronomical data in $\mathbb{R}^D$, with $D = 2$ in the case of images and $D = 3$ in the case of spectroscopic cubes.

Astronomical images in general are huge, and the part of the emission which corresponds to signal might be very diluted within the image. Because of that, we do not work with the entire images as they come. From all the available pixels (or voxels), we only use those over a *base level* intensity value, forming a subset that we call *usable pixels* set. This threshold is set as a multiple of the RMS of the noise.

Since each input data has different scales in its coordinate system (pos-pos in 2D images and pos-pos-freq in 3D cubes), we standardize it by mapping each axis to the [0, 1] range. Additionally, the intensity values are also mapped to the [0, 1] interval. In this way all the input data is in the same dimensions scales.

Even though the input data is a discrete array, we use the observed data as if it were a continuous function $f_0 : [0,1]^D \to [0,1]$, where f_0 is built as a linear interpolator in the regular grid [*].

4.2 Variational Formulation

Let $f : \Omega \subseteq [0,1]^D \to [0,1]$ be the scalar function which represents the real image (unknown and noise-free), and let $f_0 : \Omega \subseteq [0,1]^D \to [0,1]$ be the observed image, then:

$$f_0(\mathbf{x}) = f(\mathbf{x}) + \epsilon,$$

where $\mathbf{x} \in \Omega$ is an image coordinate, and ϵ is a random variable representing background noise. Since background noise is additive and always positive in astronomy, ϵ is restricted to only positive values. Then we can state the problem as a minimization of the following functional,

$$\begin{aligned} u^* = \arg\min_u J(u) &= \arg\min_u \int_\Omega L(u, u_x, u_y, \cdots)\, d\Omega \\ &= \arg\min_u \int_\Omega \underbrace{(f_0 - u)^2}_{\text{Similarity}} + \alpha \underbrace{\Psi(u - f_0)}_{\text{Domain}}\, d\Omega, \end{aligned} \tag{4.1}$$

where u^* is the unknown minimizer function, $\alpha \in \mathbb{R}_0^+$ is a weight parameter and Ψ a continuous function that we describe in Section 4.5. The Lagrangian L in this case is defined from $\mathbb{R} \to \mathbb{R}$, and its structure determines the properties of the solution we are looking for. The proposed Lagrangian has the two main terms described as follows:

- *Similarity:* this is the standard term that ensures the similarity between the observed data f_0 and the continuous approximation u used in least squares.

- *Domain:* this represents the problem domain restrictions. This have two components: 1) *flux addition*, which penalizes u when it adds extra (nonexistent) flux as part of the solution, i.e.,

[*]The linear interpolation is build with the `RegularGridInterpolator` routine from SciPy.

u must be upper bounded by f_0; and 2) *noise reduction*, which pushes u to be as similar as possible to f (real image), so the additive noise ϵ must be reduced [†].

Following the approach of the *Calculus of Variations* (Gupta, 1996), we can move from the integral formulation to the differential formulation, through the *Euler-Lagrange* equation,

$$\frac{\partial L}{\partial u} - \sum_{k=1}^{D} \frac{d}{dx_k} \frac{\partial L}{\partial u_{x_k}} = 0, \quad \text{with BC } \; u(\partial\Omega) = f(\partial\Omega), \tag{4.2}$$

where D is the dimensionality of the input data, and Ω is the domain of f_0 and u. We imposed Dirichlet boundary conditions over $\partial\Omega$. If we replace the Lagrangian in (4.1) into the Euler-Lagrange equation (4.2), we get the equation:

$$2(f_0 - u) + \alpha\Psi'(u - f_0) = 0. \tag{4.3}$$

Since partial derivatives of u are not present in the Lagrangian L, equation (4.3) is a nonlinear but algebraic equation in u. It is important to point out that although $u = f_0$ is in fact the exact solution of (4.3), we are actually looking for f, the noise-free image.

To highlight this, we write the left hand side of the last equation as $2(f+\epsilon-u)+\alpha\Psi'(u-f-\epsilon) = 0$ and point out that in the ideal case, u captures the real signal f without adding any noise, i.e. $u = f$. This equation reduces to $2\epsilon + \alpha\Psi'(-\epsilon) = 2\epsilon = O(\epsilon)$. Thus, we actually want the obtained solution not to solve equation (4.2) exactly, but approximately.

To achieve this we are restricting u to be in a particular space of functions, and then u is determined as the function in this space that minimizes (4.3). In the following section this space is described in detail.

4.3 Solution Structure

We restrict the space of possible solutions u to linear combinations of Gaussian functions. Each Gaussian function is defined by its weight, center and covariance matrix: $(c_i, \boldsymbol{\mu}_i, \Sigma_i)_{i=1}^{N}$. Then the

[†]Usually in image processing through variational calculus, the noise removal task is done by adding the term $\Psi(|\nabla u|^2)$ in the Lagrangian (smoothing the solution), but it did not show good empirical results in this approach. Instead we use the fact that ϵ is additive and we propose to handle this in the domain restrictions. This is explained in detail in Section 4.5

solutions has the form:

$$\begin{aligned} u(\mathbf{x}) &= \sum_{i=1}^{N} c_i \; \phi(\mathbf{x};\; \boldsymbol{\mu}_i, \boldsymbol{\theta}_i, \Sigma_i) + u_0 \\ &= \sum_{i=1}^{N} c_i \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T \Sigma_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)\right) + u_0, \end{aligned} \tag{4.4}$$

where the value u_0 corresponds to the *base level* introduced in Section 4.1. This is important because: 1) It is the threshold value that defines the usable pixels, 2) It does the search for the minimizer u^* numerically more stable, by avoiding having sharp Gaussians as part of the approximation (the mixture just have to approximate $f_0 - u_0$).

In order to get a simple and computationally tractable solution space, the following restrictions are imposed over u:

1. The weights c_i should be always in $\mathbb{R}_0^+$. This is for two reasons: 1) Since the Gaussians represent *pieces* of a source, they should be positive in order to ensure physical interpretation, and 2) the Gaussian agglomeration step presented in Section 5.1 uses each Gaussians function as if they were *probability density functions*, and then they should be strictly positive. This is done by defining $c_i = \hat{c}_i^{\,2}$ with $\hat{c}_i \in \mathbb{R}$.

2. The centers $\boldsymbol{\mu}_i$ of the Gaussian functions are restricted to be in the interior domain $\Omega - \partial\Omega$. This is because we only want to approximate well $f(\mathbf{x})$ in the regions with values over the *base level* μ_0. To satisfy that, we introduce the *reference centers* $\hat{\boldsymbol{\mu}}_i$, the neighborhood ratio $\boldsymbol{\delta}_i \in \mathbb{R}^D$ and a control parameter $\boldsymbol{\theta}_i \in \mathbb{R}^D$. Then the *real centers* of each Gaussian are computed as $\boldsymbol{\mu}_i = \hat{\boldsymbol{\mu}}_i + \boldsymbol{\delta}_i \odot \sin(\boldsymbol{\theta}_i)$ [†]. The solution has the final form:

$$u(\mathbf{x}) = \sum_{i=1}^{N} \hat{c}_i^{\,2} \exp\left(-\frac{1}{2}(\mathbf{x} - [\hat{\boldsymbol{\mu}}_i + \boldsymbol{\delta}_i \odot \sin(\boldsymbol{\theta}_i)])^T \Sigma_i^{-1} (\mathbf{x} - [\hat{\boldsymbol{\mu}}_i + \boldsymbol{\delta}_i \odot \sin(\boldsymbol{\theta}_i)])\right) + u_0. \tag{4.5}$$

3. The Gaussians are radially symmetric, i.e., Σ_i is a diagonal matrix with the same value in the diagonal. Additionally we introduce the *minimal and maximal width* ($\sigma_{\min}$ and $\sigma_{\max}$ respectively) parameters, which are the minimal and maximal standard deviation allowed for

[†]Here sin() is applied element-wise and $\odot$ denote element-wise multiplication.

a Gaussian. The structure of the covariance matrix is:

$$\Sigma_i = \begin{pmatrix} \sigma_i^2 & 0 & 0 \\ 0 & \sigma_i^2 & 0 \\ 0 & 0 & \sigma_i^2 \end{pmatrix}, \text{ with } \sigma_i^2 = (\sigma_{\max}^2 - \sigma_{\min}^2)\tanh(\hat{\sigma}_i^2) + \sigma_{\min}^2. \tag{4.6}$$

This is a standard way for bounding the parameters in an *unrestricted nonlinear optimization problems* (James and Roos, 1975). The reasons for bounding the standard deviation of the Gaussians are basically the followings: 1) Empirically, it makes the optimization process more numerically stable, and 2) It contributes to get a more homogeneous mixture of Gaussians, which is a good property for the agglomeration step described later in Section 5.1.

Restriction (i) is needed to ensure physical and astronomical interpretation. Restrictions (ii) and (iii) are for simplicity in the optimization step. It is also possible to extend Σ_i to have elliptical symmetry and different spatial orientation. The trade-off between model complexity and optimization cost is discussed later in Section 4.8.

4.4 Parameterization of the space of solutions

Replacing equation (4.4) into (4.2) gives us a nonlinear system of equations in the parameters of (4.5). In order to reduce the size of the solution space, we set some of these parameters heuristically:

1. *Minimal and maximal width.* These two $(\sigma_{\min}, \sigma_{\max})$ are estimated as:

$$\sigma_{\min} = \kappa \frac{1}{6D} \sum_{d=1}^{D} \mathrm{pl}_d, \tag{4.7}$$

$$\sigma_{\max} = K\sigma_{\min} \text{ with } K = \frac{N_{\text{usable pixels}}}{N_{\text{gaussians}}},$$

where D is the dimensionality of the input data, pl_d is the pixel length in each dimension, and κ a constant value. The reason we set $\sigma_{\min}$ this way is that the smallest Gaussian fits in κ pixels. For a single pixel ($\kappa = 1$) the coverage of a Gaussian (we consider 3σ) is set to be half the size of a pixel, where the pixel size is estimated as the mean of the pixel lengths in each dimension pl_d. This give us the equation (4.7). The value κ is estimated as BSize/ARes, the quotient between the *Beam Size* and *Angular Resolution* of the input data. The value of $\sigma_{\max}$ is just K times $\sigma_{\min}$, where K is an estimation of the amount of pixels each Gaussian must cover. It considers Gaussian centers uniformly distributed across the usable pixels.

2. *Reference centers.* The vectors $\hat{\boldsymbol{\mu}}_i$ are determined with the method presented next in Section 4.6.1.

3. *Neighborhood of reference centers.* The length of the radius around the actual centers can move from the reference centers, namely $\boldsymbol{\delta}_i$, is set equal for all Gaussians. We set this parameter as $\boldsymbol{\delta}_i = [\sigma_{\min}\ \sigma_{\min}]^T$ or $\boldsymbol{\delta}_i = [\sigma_{\min}\ \sigma_{\min}\ \sigma_{\min}]^T$ (depending if $D = 2$ or $D = 3$), i.e., centers can move a distance equal to the minimal width.

In this way, we reduce the parameters to be found for each Gaussian to $(\hat{c}_i, \hat{\sigma}_i, \boldsymbol{\theta}_i)_{i=1}^N$. For simplicity we define the *parameters vector*: $\boldsymbol{\eta} = [\hat{c}_1, \ldots, \hat{c}_N, \hat{\sigma}_1, \ldots, \hat{\sigma}_N, \boldsymbol{\theta}_1^T, \ldots, \boldsymbol{\theta}_N^T]$, that includes all the parameters we are looking for, and that completely parameterize our possible solutions.

4.5 Domain Restrictions

As stated previously, we have two domain restrictions: do not add extra (nonexistent) flux, and reduce the background noise from f_0, which are handled with a single term in (4.2). They are correlated under our approach, in the sense that if a single Gaussian from (4.4) fits a noisy peak caused by the positive background noise, it will add flux in the neighborhood of such peak (if $\sigma_{\min}$ is set greater than the noise resolution). Therefore, the **no-flux addition** restriction also helps to reduce the background noise from the solution. With this in mind, the expected response of $\Psi(x)$ is to not penalize negative inputs ($x < 0$), since in that case there is no extra flux addition from u. And for positive inputs ($x \geq 0$), it is expected a high and rapidly increasing penalization. Then the proposal is to use $\Psi(x) = \widehat{\psi}(\lambda\, x)$, with ψ is defined as:

$$\widehat{\psi}(x) = \begin{cases} 0 & , x < 0 \\ 10x^3 - 15x^4 + 6x^5 & , x \in [0, 1] \\ 1 & , x > 1, \end{cases} \tag{4.8}$$

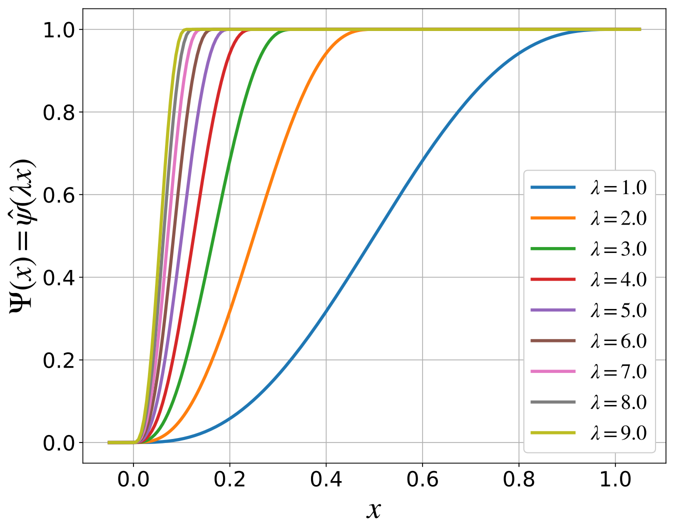


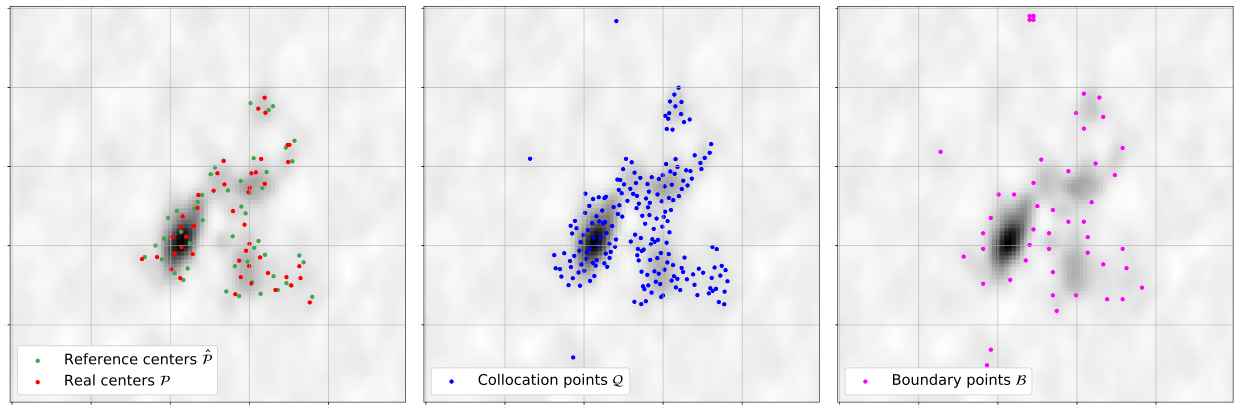
Figure 4.1: 5th order spline penalizing function and its behavior as we move λ in the range $[1., \ 9.]$. To achieve a faster penalization in the $[0., \ 1.]$ domain, higher λ values should be used.

where the definition of ψ in the $[0, 1]$ range is a 5th-order spline, built to produce continuity and smoothness in the entire domain of ψ. Because of its polynomial structure, evaluating the function and its derivatives is computational fast, since they are lower order polynomials.

In this way the parameter α controls the *maximum amount of penalization*, whereas λ controls the *speed of penalty*. To set λ, we notice that the maximum amount of penalization is reached at $x = 1/\lambda$. Since typically $|u - f| \ll 1$ during the optimization process, then $\lambda \geq 1$, with typical values in the range $[1, 10]$, as shown in Figure 4.1.

4.6 Points Generation

It is necessary to introduce three sets of points that we use, the **center points** $\mathcal{P}$ corresponding to the center of Gaussians, the **collocation points** $\mathcal{Q}$ which are the points where the equation (4.3) is evaluated, and the **boundary points** $\mathcal{B}$ which are the points at the boundary of the structures to be represented. In what follows we present the methods used to generate each one of these points.



(a) Center points. **(b)** Collocation points. **(c)** Boundary Points.

Figure 4.2: Points generation example over Orion KL image from ALMA Science Verification archive. (a) 50 *reference* and *real* center points generated as generalized Halton sequences, (b) 200 collocation points generated also as a generalized Halton sequence, and (c) 50 boundary points.

4.6.1 Center Points

As introduced in section 4.3, there are two kind of center points (Gaussian centers): The *reference center* points $\hat{\mathcal{P}}$ which are the fixed points around which the *real center* points $\mathcal{P}$ can move. An important property of these points is that *center points should be well distributed at the interior domain of the usable pixels set*. A good distribution of such points give us the capacity to get an accurate representation of the sources present in these regions.

The proposed method consist in the use of the Generalized Halton Sequences (Faure and Lemieux, 2009), which is a method for low-discrepancy sequence of N-dimensional points generation. It is a *quasi random* sequence generator that maximizes the distance between the generated points.

Then the *reference center* points $\hat{\mathcal{P}}$ are generated as a sequence of Halton Points over the regions of usable pixels. Since the regions of usable pixels are very irregular (can have any shape), and Halton sequences fills square domains ($[0,1]^D$), then we generate Halton points until our irregular domain is filled with the desired number of points.

It is important to mention that the real center points $\mathcal{P}$ are the same as the fixed center points $\hat{\mathcal{P}}$ in the initial conditions of the optimization, i.e., with values $\boldsymbol{\theta}_i = \mathbf{0}$ for $i \in [1, N]$. An example of how these point are generated is shown in Figure 4.2a.

4.6.2 Collocation Points

These correspond to the points in the interior domain $\Omega - \partial\Omega$ where the equation (4.3) is evaluated. Because of the need to get a determined (or overdetermined) nonlinear system of equations, there is a greater amount of these points than the center points, more precisely they satisfy $|\mathcal{Q}| \geq (D+2)|\mathcal{P}| - |\mathcal{B}|$. In the same way as center points, *collocation points must be well distributed across the entire interior domain of usable pixels set*: With a set of points with a good coverage of the interior domain, we force the equation (4.3) to be minimized across the regions of usable pixels.

These are also generated as Halton Sequences points in the interior domain. Moreover, collocation point include the *reference center* points previously generated: $\hat{\mathcal{P}} \subset \mathcal{Q}$, and add the remaining points to get a consistent system. An example of how these point are generated is shown in Figure 4.2b.

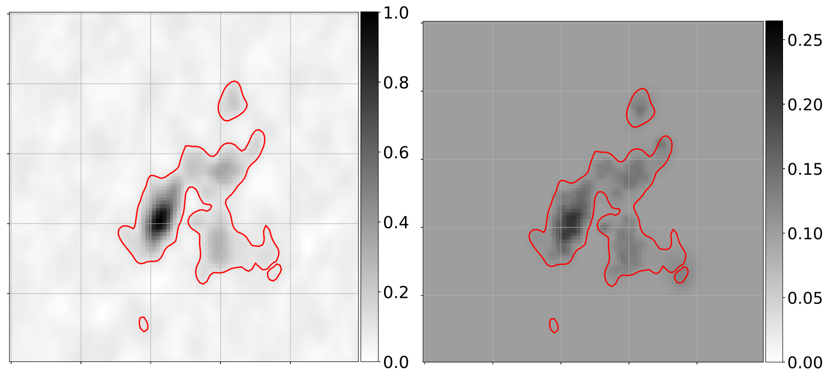
The idea of $\hat{\mathcal{P}} \subset \mathcal{Q}$ is that each Gaussian has a collocation point near its center. Having a collocation point close to each Gaussian center it is necessary to have a good solution, otherwise, it has been observed empirically that a single Gaussian could grow too much deteriorating the solution.

4.6.3 Boundary Points

In order to evaluate the boundary condition $u(\partial\Omega) = f(\partial\Omega)$, a finite set of points in the boundary are selected. These points are generated as a random sample in the set of boundary pixels. The random sampling is performed assuming a uniform probability on the boundary pixels. We iteratively sample the boundary pixels, and every time a pixel is selected, this and its neighbors pixels are set with probability 0. This is done with the aim to get a *spatially distributed* sample. An example of how these points are generated is show in Figure 4.2c

4.7 Initial Guess Estimation

Once the *reference centers* and initial *real centers* are set, the next step is to estimate an initial guess for the weight and width parameters of the Gaussians: (c_i, σ_i) for $i \in [1, N]$. The main goal is to generate an initial guess $\boldsymbol{\eta}^{(0)}$ that do not induce an *strong bias* towards a particular solution in the space of parameters $\boldsymbol{\Gamma}$. This is achieved with an homogeneous configuration of such parameters. For



(a) Original. **(b)** Initial Guess Estimation.

Figure 4.3: (a) Original Orion KL image from ALMA Science Verification data set, and (b) its initial guess representation through the method described in the present section.

this purpose the method described below is used:

1. For each center $\boldsymbol{\mu}_i$ in $\mathcal{P}$ we compute the distance to its nearest neighbor center. We consider this distance as the initial σ_i estimation.

2. For each center $\boldsymbol{\mu}_i$ we compute a neighborhood $\mathcal{N}_i$ of centers which are at a radius of $3\sigma_i$. The Gaussians in this neighborhood are considered as making a flux contribution in the region covered by the i−th Gaussian.

3. Then the estimated initial weight is $c_i = f(\boldsymbol{\mu}_i)/(|\mathcal{N}_i| + 1)$, i.e., the intensity observed at $\boldsymbol{\mu}_i$ it is distributed between all the Gaussians in the neighborhood $\mathcal{N}_i$ (including the Gaussian centered at such point).

4. In case that a center $\boldsymbol{\mu}_i$ has no neighbors, then the initial values are is as: $c_i = f(\boldsymbol{\mu_i})$ and $\sigma_i = \sigma_{\min}$, i.e., this solitary Gaussian covers its region by its own.

The proposed method produces an homogeneous initial guesses in both $\mathbf{c}$ and $\boldsymbol{\sigma}$ as shown in Figure 4.3. Note that it captures the morphology of the structure of the data.

4.8 Numerical Optimization

To complete the first step of the method of obtaining a continuous approximation of $f(\mathbf{x})$, we need to find the *parameters vector* $\boldsymbol{\eta} = [\hat{c}_1, \ldots, \hat{c}_N, \hat{\sigma}_1, \ldots, \hat{\sigma}_N, \boldsymbol{\theta}_1^T, \ldots, \boldsymbol{\theta}_N^T]$ for the minimizer function $u^*(\mathbf{x})$. For doing this we define $E(\mathbf{x},\boldsymbol{\eta})$ as the expression obtained when we evaluate the proposed solution (4.5) in the left side of the Euler-Lagrange equation (4.2) at $\mathbf{x}$ and for a given $\boldsymbol{\eta}$:

$$\begin{aligned} E(\mathbf{x},\boldsymbol{\eta}) =& 2(f_0(\mathbf{x}) - u(\mathbf{x};\boldsymbol{\eta})) + \alpha\Psi'(u(\mathbf{x};\boldsymbol{\eta}) - f_0(\mathbf{x})), \quad \text{with} \\ u(\mathbf{x};\boldsymbol{\eta}) =& \sum_{i=1}^{N} \hat{c}_i^{\,2} \exp\left(-\frac{1}{2}(\mathbf{x} - [\hat{\boldsymbol{\mu}}_i + \boldsymbol{\delta}\odot\sin(\boldsymbol{\theta}_i)])^T \Sigma_i^{-1}(\mathbf{x} - [\hat{\boldsymbol{\mu}}_i + \boldsymbol{\delta}\odot\sin(\boldsymbol{\theta}_i)]\right) + u_0, \end{aligned} \tag{4.9}$$

and also we define $\mathrm{BC}(\mathbf{x},\boldsymbol{\eta}) = u(\mathbf{x};\boldsymbol{\eta}) - f_0(\mathbf{x})$. Then the problem can be as solving

$$E(\mathbf{x},\boldsymbol{\eta}) = 0 \text{ with boundary condition } \mathrm{BC}(\mathbf{x},\boldsymbol{\eta}) = 0.$$

The nonlinear system of equations arises by evaluating $E(\mathbf{x},\boldsymbol{\eta})$ at the collocation points, and evaluating $\mathrm{BC}(\mathbf{x},\boldsymbol{\eta})$ at the boundary points. Since for each Gaussian function in $u(\mathbf{x};\boldsymbol{\eta})$ there are $D+2$ parameters, then we need at least $(D+2)\cdot N$ of such evaluations. If $\mathbf{x}_{col}^{(i)} \in \mathcal{Q}$ are the selected collocation points ($|\mathcal{Q}| = Q$) and $\mathbf{x}_{bound}^{(i)} \in \mathcal{B}$ are the selected boundary points ($|\mathcal{B}| = B$), such that $|\mathcal{Q}\bigcup\mathcal{B}| \geq (D+2)\cdot N$, then the following nonlinear system of equations is obtained:

$$r(\boldsymbol{\eta}) = \begin{pmatrix} E\left(\mathbf{x}_{col}^{(1)},\boldsymbol{\eta}\right) \\ \vdots \\ E\left(\mathbf{x}_{col}^{(Q)},\boldsymbol{\eta}\right) \\ \mathrm{BC}\left(\mathbf{x}_{bound}^{(1)},\boldsymbol{\eta}\right) \\ \vdots \\ \mathrm{BC}\left(\mathbf{x}_{bound}^{(B)},\boldsymbol{\eta}\right) \end{pmatrix} = \mathbf{0}, \tag{4.10}$$

which is determined or overdetermined by construction. There are two possible ways to solve this problem:

1. Directly solve $r(\boldsymbol{\eta}) = \mathbf{0}$ for $\boldsymbol{\eta}$ with the Newton method or one of its variants. However it has the drawback that only works when the system (4.10) is determined (not overdetermined).

2. Instead we can define a residual function $R(\boldsymbol{\eta}) = \frac{1}{2}r(\boldsymbol{\eta})^T r(\boldsymbol{\eta})$ and try to minimize it, which is the least squares approach. This works well for both, determined and overdetermined systems.

In many cases we need a fine-grained evaluation of $E(\mathbf{x};\boldsymbol{\eta})$ in the domain Ω to get an accurate approximation, producing overdetermined systems. Additionally we observed a good empirical behavior in the nonlinear least squares solvers for this problem. Because of these reasons we choose the least squares minimization approach. For the minimization we use the Levenberg-Marquardt algorithm (LM) (Moré, 1978). We choose LM because of its stability properties (local convergence ensured) and its computational efficiency. Because of the complex structure of the space of solutions of $\boldsymbol{\eta}$, we think that minimizing the residual is much more tractable than trying to find a root.

LM works by iteratively solving the following least squares problem:

$$\begin{gathered}\left(J_{\eta}^{T} J_{\eta} + \lambda \operatorname{diag}\left(J_{\eta}^{T} J_{\eta}\right)\right) \boldsymbol{\delta} = J_{\eta}^{T} r(\boldsymbol{\eta}^{s}), \\ \boldsymbol{\eta}^{(s+1)} = \boldsymbol{\eta}^{(s)} + \boldsymbol{\delta},\end{gathered} \tag{4.11}$$

where s is the current iteration step, J_{η} is the Jacobian matrix of $r(\boldsymbol{\eta})$ and λ is the *damping parameter*. The parameter λ controls the convergence behavior between Newton-Gauss Method (slow residual reduction but stable) and Steepest Gradient Descent (fast residual reduction but prone to fall in local minimum).

The LM algorithm implementation used corresponds to the SciPy's version which in turn calls a wrapper over least-squares algorithms implemented in MINPACK (lmder, lmdif). The implementation is based on paper (Moré, 1978), and it is very robust and efficient with a lot of smart tricks.

4.8.1 Dimensionality and Computational Complexity

A reasonable way to measure the computational complexity of the method, is by the number of Gaussian function evaluation required to compute the residual vector, define in Equation (4.10). Let N be the number of Gaussians in the mixture representation, and γ the number of free parameters of each Gaussian ($D + 2$ in our model). Every time we evaluate $E\left(\mathbf{x}_{col}^{(i)}, \boldsymbol{\eta}\right)$, N Gaussian function evaluations are performed. Additionally the number of *collocation points* plus the number of *boundary points* (which is equal to $|r(\boldsymbol{\eta})|$) is $\geq \gamma N$. Then the *computational complexity* is $O(\gamma N^2)$. It means that it is linear ($O(\gamma)$) in the number of free parameters, and quadratic ($O(N^2)$) in the number of Gaussians.

Considering that $r(\boldsymbol{\eta})$ is evaluated a several times in the numerical minimization of LM, in order to have reasonable computing times, it is necessary to keep the computation of it fast. Adding some other free parameters, like a sparse covariance matrix Σ_i for each Gaussian will make it slower, but it will be worst be to have a representation with too many Gaussians.

5 | Second Step: Gaussian Mixture Reduction

5.1 Gaussian Agglomeration

Now that we have a continuous representation u of the astronomical image as a mixture of Gaussians, the next step is to use this representation to extract information about the sources that are present and its hierarchical relationships. For this purpose we use the key fact that $u(\mathbf{x})$ is a linear combination of Gaussian functions, and perform GMR algorithms over it. This was the main reason of why we use it instead of any other particular function.

We first need to define the *Gaussian Mixture* as a linear combination of normal probability density functions:

$$\begin{aligned} h_N(\mathbf{x}) &= \sum_{i=1}^{N} w_i \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \Sigma_i) \quad \text{with} \\ \mathcal{N}(x; \boldsymbol{\mu}_i, \Sigma_i) &= \frac{1}{\sqrt{(2\pi)^D |\Sigma_i|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \Sigma_i^{-1} (\mathbf{x}-\boldsymbol{\mu}_i)\right), \end{aligned} \tag{5.1}$$

that we also denote like $\tilde{M} = \{(w_1, \boldsymbol{\mu}_1, \Sigma_1), (w_2, \boldsymbol{\mu}_2, \Sigma_2), \ldots, (w_N, \boldsymbol{\mu}_N, \Sigma_N)\}$, where w_i are the weights, $\boldsymbol{\mu}_i$ are the mean vectors, and Σ_i the covariance matrices. Note that if $\sum_{i=1}^{N} w_i = 1$, then this Gaussian mixture is also a *probability density function* (PDF). The GMR algorithms consist of reducing an N-components Gaussian Mixture $h_N(\mathbf{x})$ to an M-components Gaussian mixture $h_M(\mathbf{x})$ (with $M < N$), such that h_M and h_N are the least possible dissimilar, in a sense to be defined next in this section.

Since all the results in the literature of GMR apply only on Gaussian mixtures as presented in

(5.1), we need to do some manipulation over the $u(\mathbf{x})$ function:

$$\begin{aligned} u(\mathbf{x}) - u_0 &= \sum_{i=1}^{N} c_i \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right) \\ &= \sum_{i=1}^{N} \underbrace{c_i \sqrt{(2\pi)^D |\Sigma_i|}}_{w_i} \frac{1}{\sqrt{(2\pi)^D |\Sigma_i|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right) \\ &= \sum_{i=1}^{N} w_i \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \Sigma_i). \end{aligned} \tag{5.2}$$

We need to point that (5.2) is the same $u(\mathbf{x})$ but rewritten and interpreted in a convenient way for GMR purposes. Each component in this mixture, represent exactly the same component in the original linear combination of Gaussians. Moreover, the only thing that changes in (5.2) is the weight coefficient: Gaussian functions with greater width, will have a relatively greater w_i coefficient in the Mixture representation (and vice versa).

There are many different algorithms to perform GMR, and a complete survey of them can be found in Crouse et al. (2011). Between these algorithms we choose Runnalls (2007)'s approach because of the following reasons:

1. It is computationally efficient.
2. It is the most robust among simple GMR algorithms: Advanced algorithms like GMRC (Gaussian Mixture Reduction via Clustering) (Schieferdecker and Huber, 2009) and COWA (Constraint Optimized Weight Adaptation) (Chen et al., 2010), use the result of Runalls algorithm for initial guess estimation.
3. It allows us to build a hierarchical binary tree describing the relations between the Gaussian components and the sources in formation.

In order to understand how this algorithm works, we need to introduce the following two concepts:

1. The **moment preserving merge** of two Gaussian components $\{(w_i, \boldsymbol{\mu}_i, \Sigma_i), (w_j, \boldsymbol{\mu}_j, \Sigma_j)\}$ corresponds to a single Gaussian $(w_m, \boldsymbol{\mu}_m, \Sigma_m)$ which preserves the zeroth, first and second-order

moments of the two-components mixture. Its parameters are obtained as follows:

$$\begin{aligned} w_m &= w_i + w_j \\ \boldsymbol{\mu}_m &= \frac{w_i}{w_m}\boldsymbol{\mu}_i + \frac{w_j}{w_m}\boldsymbol{\mu}_j \\ \Sigma_m &= \frac{w_i}{w_m}\Sigma_i + \frac{w_j}{w_m}\Sigma_j + \frac{w_i w_j}{w_m^2}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T . \end{aligned} \tag{5.3}$$

2. Runnalls (2007)'s approach compares each two components mixture $\{(w_i, \boldsymbol{\mu}_i, \Sigma_i), (w_j, \boldsymbol{\mu}_j, \Sigma_j)\}$ with its moment preserving merge $\{(w_m, \boldsymbol{\mu}_m, \Sigma_m)\}$ using a **Kullback-Liebler divergence** (KLd) upper bound. In fact because KLd compares PDFs, it actually is applied over the normalized version of both:

$$\{(w_i/(w_i + w_j), \boldsymbol{\mu}_i, \Sigma_i),\ (w_j/(w_i + w_j), \boldsymbol{\mu}_j, \Sigma_j)\} \text{ and } \{(1, \boldsymbol{\mu}_m, \Sigma_m)\} \text{ respectively.}$$

   In Williams and Maybeck (2003), the KLd it is described as the "ideal cost function" for Gaussian mixture reduction , but it has the disadvantage that has no *closed-form* in the cases of Gaussian mixtures. However Runnalls (2007) obtained the following upper bound:

$$d_{KL}(i, j) = \frac{1}{2}\Big((w_i + w_j)\log|\Sigma_m| - w_i \log|\Sigma_i| - w_j \log|\Sigma_j|\Big), \tag{5.4}$$

   which exhibit very good empirical results.

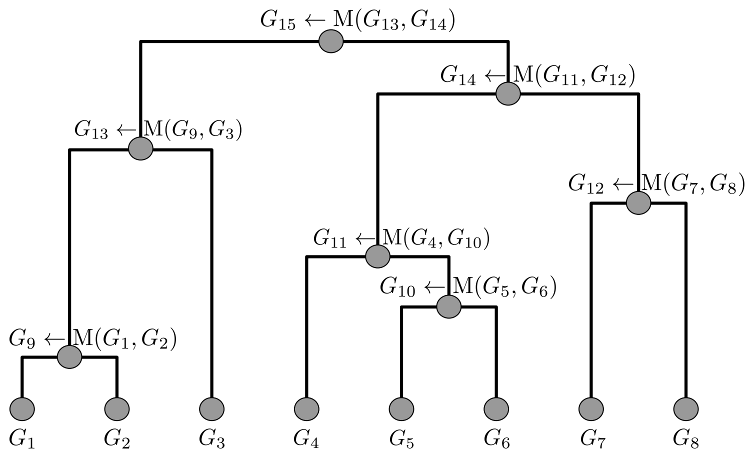
Then the agglomeration and creation of the hierarchical tree $\mathcal{T}$ is as presented in Algorithm 1. In simple words it works as follows:

> *While more than* 1 *components remain in the Gaussian mixture* $\tilde{M}$*, find the two components that are less dissimilar in the sense of* (5.4)*, and replace them in* $\tilde{M}$ *with its moments preserving merge.*

In this way we build a binary tree $\mathcal{T}$ describing the hierarchical relations between the components that represents the sources. A visualization of the creation process of the tree is shown in Figure 5.1. This tree is built from the bottom to top and each node represent a single Gaussian. Every time we merge two components, it is represented with the merge of two branches into a single node. This node contains the moment preserving merge of the Gaussians representing such branches.

Algorithm 1 Generation of the hierarchical tree $\mathcal{T}$ through Runalls' GMR

```
1: $\mathcal{T} \leftarrow$ *empty tree*
2: $\tilde{M} \leftarrow [(w_1,\mu_1,\Sigma_1),\ldots,(w_N,\mu_M,\Sigma_M)]$
3: for $i = 1$ : length($\tilde{M}$) do
4:     $\mathcal{T}$.addnode($\tilde{M}[i]$)
5: while length($\tilde{M}$) $\neq 1$ do
6:     min = +inf
7:     for $i = 1$ : length($\tilde{M}$) do
8:         for $j = i+1$ : length($\tilde{M}$) do
9:             if $d_{KL}(\tilde{M}[i],\tilde{M}[j]) <$ min then
10:                $i_{\mathtt{min}} = i$; $j_{\mathtt{min}} = j$
11:    $(w_m,\mu_m,\Sigma_m) \leftarrow$ merge($\tilde{M}[i_{\mathtt{min}}],\tilde{M}[j_{\mathtt{min}}]$)
12:    $\tilde{M}$.add($(w_m,\mu_m,\Sigma_m)$)
13:    $\mathcal{T}$.addnode($(w_m,\mu_m,\Sigma_m)$)
14:    $\mathcal{T}$.node($(w_m,\mu_m,\Sigma_m)$) set as father of $\mathcal{T}$.node($\tilde{M}[i_{\mathtt{min}}]$)
15:    $\mathcal{T}$.node($(w_m,\mu_m,\Sigma_m)$) set as father of $\mathcal{T}$.node($\tilde{M}[j_{\mathtt{min}}]$)
16:    $\tilde{M}$.remove($\tilde{M}[i_{\mathtt{min}}]$)
17:    $\tilde{M}$.remove($\tilde{M}[j_{\mathtt{min}}]$)
```



$G_{15} \leftarrow \mathrm{M}(G_{13}, G_{14})$
$G_{14} \leftarrow \mathrm{M}(G_{11}, G_{12})$
$G_{13} \leftarrow \mathrm{M}(G_9, G_3)$
$G_{12} \leftarrow \mathrm{M}(G_7, G_8)$
$G_{11} \leftarrow \mathrm{M}(G_4, G_{10})$
$G_{10} \leftarrow \mathrm{M}(G_5, G_6)$
$G_9 \leftarrow \mathrm{M}(G_1, G_2)$
G_1
G_2
G_3
G_4
G_5
G_6
G_7
G_8

Figure 5.1: Example tree describing the hierarchical agglomeration process. Here G_i represent a single Gaussian, and $M()$ is the merging function which in this case computes the *moment preserving merge*.

5.2 Gaussian Decomposition

In this step we use the hierarchical tree $\mathcal{T}$ built by Algorithm 1. An instance of this tree is show in Figure 5.1. It describes the sources in the image and its hierarchical relationships. The decomposition consists of traversing $\mathcal{T}$ from top to bottom, decomposing it in a set of not overlapping subtrees, each one representing a source (See Algorithm 2). In particular we are interested in a decomposition function d that takes a set of K not overlapping subtrees of $\mathcal{T}$: $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_K\}$, and indicates which one to decompose next. Then, the function d defines a *decomposition strategy*. We show now two of such strategies:

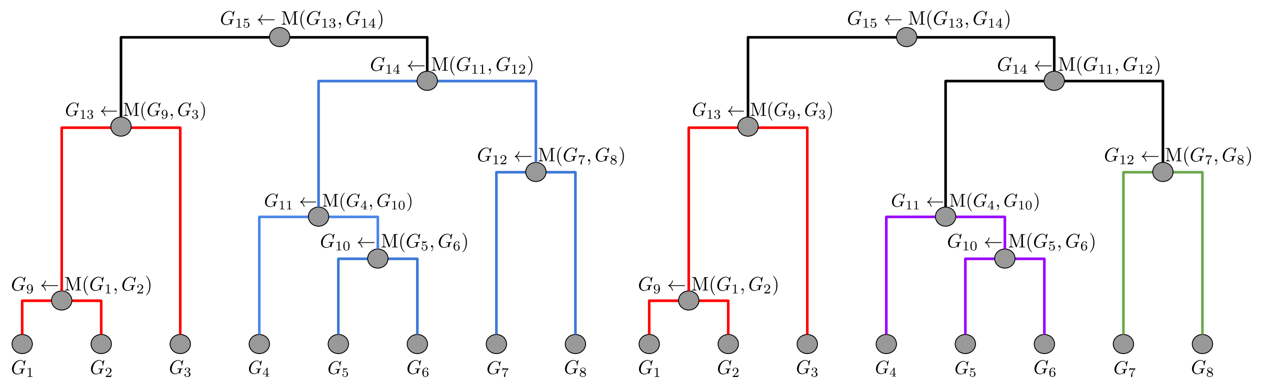
1. **Reverse order.** This corresponds to decompose the tree in the same exact order in which it was built, but in reverse. This leaves the hierarchical structure completely in the hands of the KLd bound metric.

2. **Manual decomposition.** The hierarchical tree can be decomposed by a domain expert (astronomer), iteratively choosing the subtree to decompose at each step. In this way we can analyze only the structures we are interested in.

Algorithm 2 Decomposition of $\mathcal{T}$ into S subtrees with decomposition function d.

```
1: tree_list ← [$\mathcal{T}$]
2: while len(tree_list) $\neq S$ do
3:     $i_{\min}$ ← $d$(tree_list)
4:     $\mathcal{T}_{\text{left}}, \mathcal{T}_{\text{right}}$ ← split(tree_list[$i_{\min}$])
5:     tree_list.remove($\mathcal{T}_{i_{\min}}$)
6:     tree_list.add($\mathcal{T}_{\text{left}}$)
7:     tree_list.add($\mathcal{T}_{\text{right}}$)
```

An example of the decomposition process is illustrated in Figure 5.2. Here are shown the first two decomposition steps with the *reverse order* strategy. In Figure 5.2a two sources are represented with the two subtrees (red and blue), while in figure 5.2b the “blue” subtree is decomposed into its two subtrees (cyan and green). Each of these represent a source that are hierarchical components of the source described by the blue tree. This process may continue until each subtree is a single Gaussian, which we also refer as leaf Gaussian.


$G_{15} \leftarrow \mathrm{M}(G_{13}, G_{14})$
$G_{14} \leftarrow \mathrm{M}(G_{11}, G_{12})$
$G_{13} \leftarrow \mathrm{M}(G_9, G_3)$
$G_{12} \leftarrow \mathrm{M}(G_7, G_8)$
$G_{11} \leftarrow \mathrm{M}(G_4, G_{10})$
$G_{10} \leftarrow \mathrm{M}(G_5, G_6)$
$G_9 \leftarrow \mathrm{M}(G_1, G_2)$
G_1 G_2 G_3 G_4 G_5 G_6 G_7 G_8
$G_{15} \leftarrow \mathrm{M}(G_{13}, G_{14})$
$G_{14} \leftarrow \mathrm{M}(G_{11}, G_{12})$
$G_{13} \leftarrow \mathrm{M}(G_9, G_3)$
$G_{12} \leftarrow \mathrm{M}(G_7, G_8)$
$G_{11} \leftarrow \mathrm{M}(G_4, G_{10})$
$G_{10} \leftarrow \mathrm{M}(G_5, G_6)$
$G_9 \leftarrow \mathrm{M}(G_1, G_2)$
G_1 G_2 G_3 G_4 G_5 G_6 G_7 G_8

(a) Two sources tree **(b)** Three sources tree

Figure 5.2: First two step of the decomposition process.

A key feature for describing the sources, is that when we reconstruct them, we do not use the moment preserving merge Gaussian at the root of the subtree. Instead we use the leaf Gaussians of such subtree. In this way we produce an accurate representation of each source.

6 | Experiments

This chapter is divided in two sections. Section 6.1 shows a complete *pipeline* of our method. This consists of a step-by-step explanation and graphical description of the execution flow of our algorithm, with all its technicalities. Then in Section 6.2 we perform an assessment of our proposal, verifying if the starting requirements and goals are being correctly satisfied.

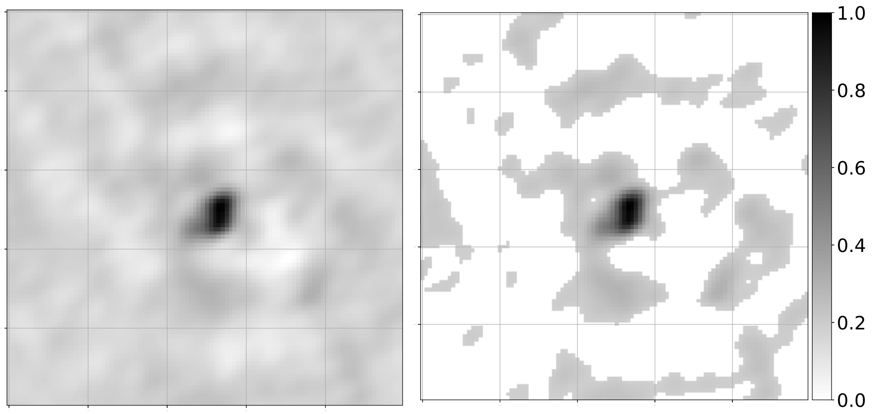
6.1 Pipeline

To illustrate the flow of execution of our method, we use the NGC3256 observation from the ALMA Science verification archive. The description of this observation is shown in Table 6.1 and can be visualized in Figure 6.1a.

The first step consists in estimating the *base level* u_0, and then apply the threshold over the original image to obtain the region of usable pixels. The value used for u_0 is the RMS of the data, which is an approximation of the RMS of the noise (the signal is very diluted within the image). The result of the thresholding is shown in Figure 6.1b.

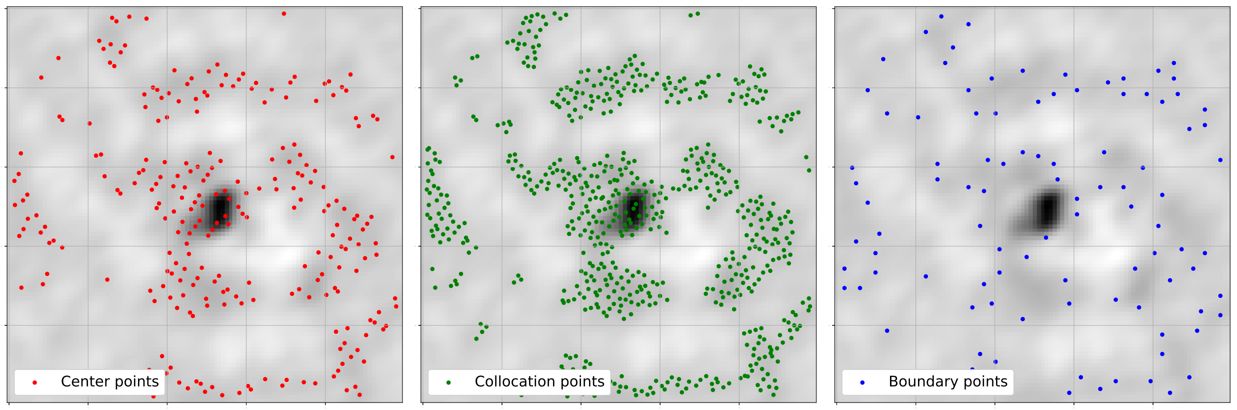
In this experiment we decided to use a mixture of 250 Gaussian components to approximate the signal. The 250 *reference center* points of the Gaussians are shown in Figure 6.2a. The number of *boundary points* was set to 100 (See Figure 6.2b), and the number of collocation points was set to 900 (Recall from Section 4.6 that $|\mathcal{Q}| \geq (D+2)|\mathcal{P}| - |\mathcal{B}|$). The *collocation points* generated can be seen in Figure 6.2c.

The next step correspond to the initial guess estimation for the *parameters vector* $\boldsymbol{\eta}$. As explained in Section 4.6.1 we set $\boldsymbol{\theta}_i = \mathbf{0}\ \forall i \in [1, N]$, and then the initial *real center* points match the *reference center* points. The estimation for c_i and σ_i is done with the method presented in Section 4.7. Figure 6.3b shows how the approximation looks like with the initial guess estimation, i.e., $u(\mathbf{x}; \boldsymbol{\eta}_0)$ evaluated



(a) Original image **(b)** Thresholded image

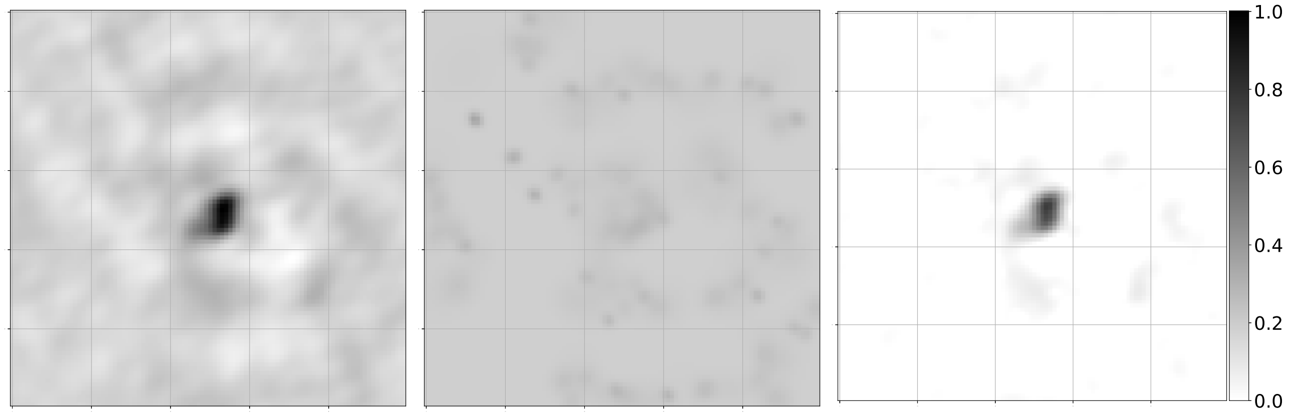
Figure 6.1: Original and thresholded NGC3256 images, showing the region of usable pixels.



(a) Center points **(b)** Collocation points **(c)** Boundary points

Figure 6.2: Center, collocation and boundary points generated over the NGC3256 image.

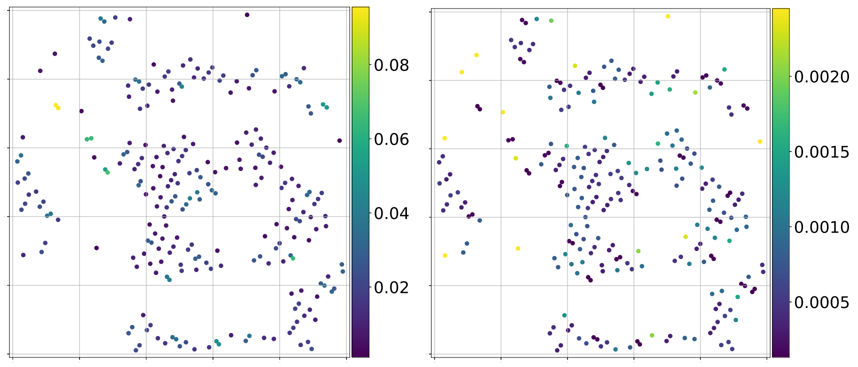
in the pixels grid. In general the initial guess values are far below the observed data f_0. This fact is shown in Figure 6.3c (which is computed as $f_0(\mathbf{x}) - u(\mathbf{x}; \boldsymbol{\eta}_0)$), were the residual remains very similar to the original image.



(a) Original image **(b)** Initial Guess **(c)** Residual

Figure 6.3: Original image, initial guess instantiation, and residual image.

For a better visualization of the initial values for the c_i and σ_i parameters, in Figure 6.4 we present a *graphical distribution* of them. Here we show the corresponding Gaussian centers, with its associated c_i (See Figure 6.4a) and σ_i^2 values (See Figure 6.4b). We observe that these parameters are approximately uniformly distributed, which was the goal of the initial guess estimation procedure.



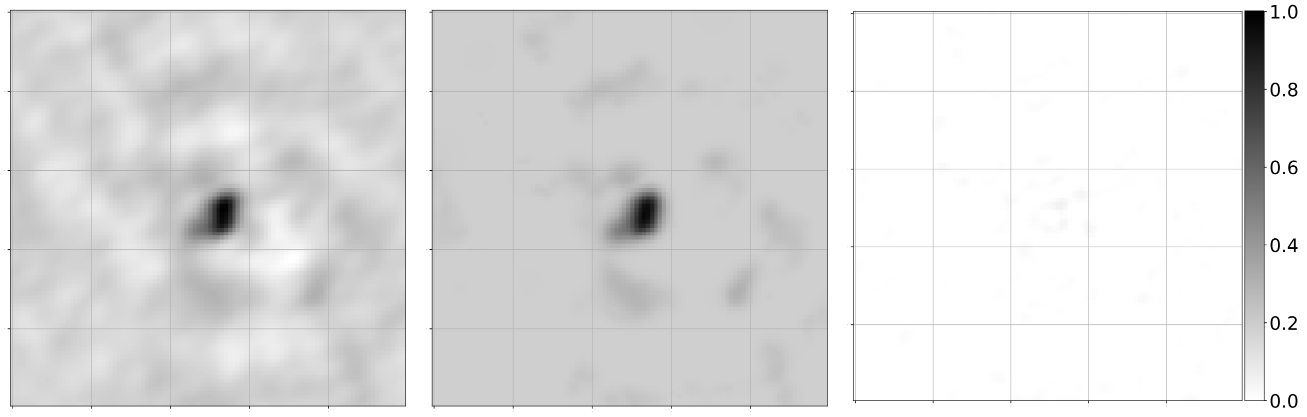
(a) Initial distribution of c_i values **(b)** Initial distribution of σ_i^2 values

Figure 6.4: Graphical distribution for the initial c_i and σ_i^2 parameters.

We want to point out that the only parameters that have to be set by the user, are the number of centers N, α and λ. All the remaining parameters are heuristically estimated from the input data. The user is able to modify these other parameters ($\sigma_{\min}$, $\sigma_{\max}$, κ, among others) but we recommend leaving them as they are. In this experiment we have set $\alpha = 1$ and $\lambda = 5$, which correspond to the default parameter values.

Once the initial guess $\boldsymbol{\eta}_0$ is estimated we are ready to run the numerical optimization, which is least squares minimization through the LM algorithm. An important LM parameter for termination criterion, is the maximum number of function evaluations. We set this parameter as $100(|\boldsymbol{\eta}| + 1)$, according the SciPy's indications [†]. The numerical optimization successfully converged to a solution, which is shown in Figure 6.5b. The termination criterion was the no reduction of the relative error between two consecutive iterations. The elapsed time of the optimization was 70.43[s] and required 22037 function evaluations of $r(\boldsymbol{\eta})$.

For comparison, we show in Figure 6.5c the residual between to original image and the obtained approximation. We observe that it is almost uniformly zero in the regions of the usable pixels. Figures 6.6a and 6.6b show the *graphical distribution* of the c_i and σ_i^2 parameters after the optimization. We remark that these parameters remain uniform across the Gaussians, with two exceptions: 1) in the pixels where there is more concentration of flux, the values c_i tend to increase, and 2) in the isolated Gaussians (no neighbors near), the values σ_i^2 tend to increase too.

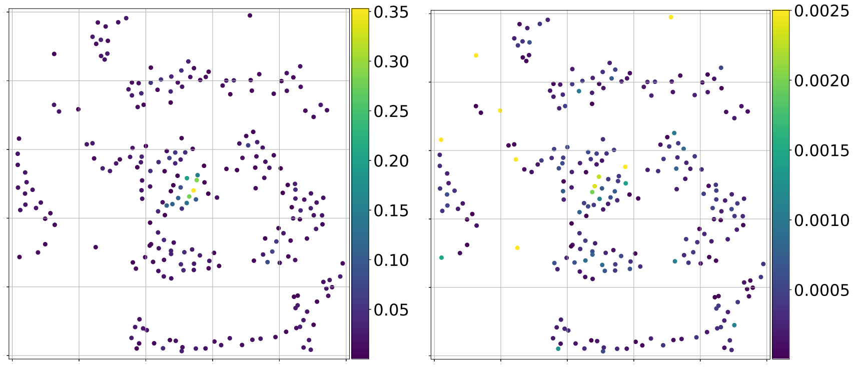


(a) Original image **(b)** Obtained solution **(c)** Residual

Figure 6.5: Original image, approximated solution, and residual image.

Once we have the mixture representation computed, we are able to run the GMR procedure (see Algorithm 1) to build the hierarchical tree $\mathcal{T}$. The way the hierarchical decomposition process works is illustrated in Figure 6.7. In this Figure we plot a single contour line at the level of u_O (base level) for each source component identified, each with a different color. We applied the

[†]scipy.optimize.least_squares documentation.

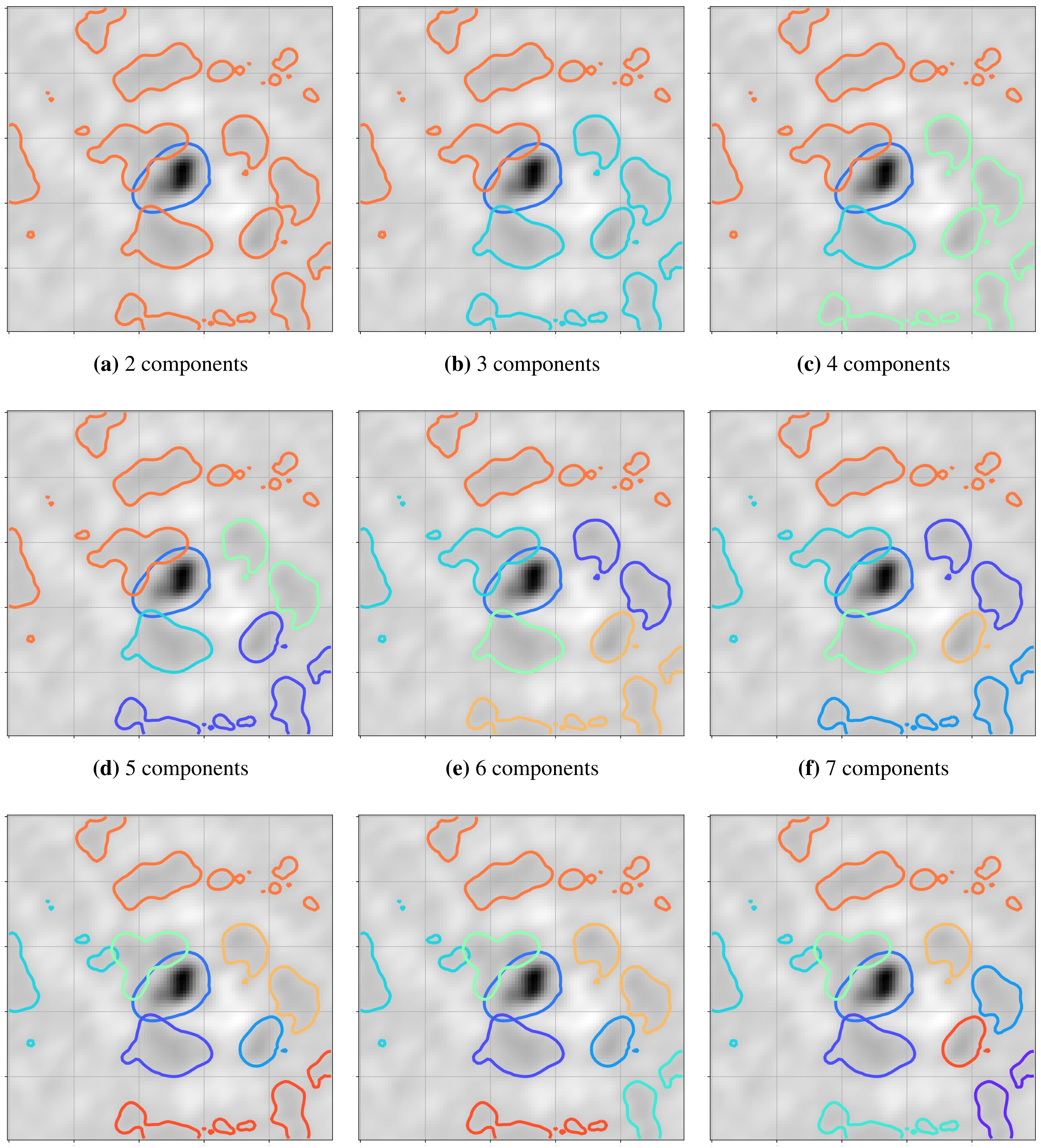


(a) Final distribution of c values **(b)** Final distribution of σ^2 values

Figure 6.6: Graphical distribution for the final c_i and σ_i^2 parameters.

manual decomposition strategy, starting with a 2-components decomposition in Figure 6.7a to the 10-components decomposition in Figure 6.7i. At each step, we arbitrarily choose the structure to decompose. We observe that the results are consistent; Even when there are disconnected sources represented as if they were one, they are spatially contiguous, and can be decomposed in successive steps.

In the next section we perform a conscientious assessment for the parameters configuration and results we have presented in this pipeline.



(a) 2 components (b) 3 components (c) 4 components

(d) 5 components (e) 6 components (f) 7 components

(g) 8 components (h) 9 components (i) 10 components

Figure 6.7: Illustration of the hierarchical decomposition of sources over NGC3256. The decomposition was done with the *manual* method, selecting at each step which source to decompose.

6.2 Experiments

In this section we study quantitatively and qualitatively the behavior of the main stages of the computational implementation of our method [*], and its dependency on the number of Gaussians N and the weight parameter α from equation (4.1). We perform experiments over a set of relevant astronomical images and spectroscopic cubes from the ALMA science verification archive (Hills, 2011). In particular we are interested in observations that have a clumpy/hierarchical structure. We have chosen this dataset because it is highly validated and studied by the astronomical community. Table 6.1 shows a summary of the images an cubes used in the experiments.

First, it is necessary to introduce the metrics we use to quantify the experiments performed next. We evaluate four measurements over the residual $(f_0 - u)$ and over the approximated solution u. Lets define $\mathcal{I}$ as the set with the indexes of the *usable pixels*, and f_{0_i}, u_i the values of $f_0(\mathbf{x})$ and $u(\mathbf{x})$ evaluated at the location of the i-th pixel. Then these measurements are:

1. **RMS of residual**: RMS value of residual over the regions of usable pixels: $\sqrt{\frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}(f_{0_i} - u_i)^2}$.
2. **Flux addition**: Added flux by the approximation: $\sum_{i\in\mathcal{I}}(u_i - f_{0_i})^+$. [†]
3. **Flux lost:** Amount of flux not represented by the approximation: $\sum_{i\in\mathcal{I}}(f_{0_i} - u_i)^+$.
4. **Sharpness:** Measurement of how sharp (not smooth) and noisy is the solution: $\sum_{i\in\mathcal{I}}|\nabla u_i|^2$. The gradient at each pixel is computed with finite forward differences.

Notice that *RMS of residual* evaluates how similar is u to f_0, i.e., evaluate the first term of the functional (4.1). The remaining measures, quantify both: if flux is being added or not represented, and the smoothness of the solution. These evaluate the second term in the functional (4.1).

6.2.1 Number of Gaussians

The first set of experiments consists in the study of the *quality* of the obtained solution u, and the number of Gaussians required for it.

[*]This implementation is based on Python and supported by standard libraries: NumPy, SciPy, Numba, Matplotlib, AstroPy, among others. All this code is publicly available in the repository for reproducibility of the experiments.

[†]Here $()^+$ denotes the *positive part function* defined as: $(x)^+ := \max(0, x)$.

Name	**RA** [pix]	**DEC** [pix]	**FREQ** [pix]	**Total** [pix]	**ARes** [$''$]	**BSize** [$''$]	**SRes** [MHz]
Orion_KL-CH3OH	100	100	1	10000	0.40	1.37	-
NGC3256-CO1_0	100	100	1	10000	1.00	4.61	-
NGC4038-CO2_1	500	500	1	250000	0.13	0.66	-
Orion_KL-CH3OH	100	100	41	41000	0.40	1.38	0.49
HD163296-CO3_2	512	512	140	36700160	0.05	0.42	0.13

Table 6.1: FITS data used from ALMA Science Verification archive. The first three entries correspond to 2D **images**, and the remaining two are 3D spectroscopic **cubes**.

In Figure 6.8 (left column) we observe the results for the images in Table 6.1. Note that there is a decreasing tendency for all the measures but sharpness. Then with a higher number of Gaussian in the mixture, we achieve a more accurate representation u. It also produces a reduction in the flux addition and flux lost. The increase in the sharpness is an expected result; while more components we add to the Gaussian mixture u, there is more capacity to represent the details in the data, and then increases the sharpness.

We also present in Figure 6.9 a computational time analysis of our method. Here we observe that the elapsed time required to complete the first step of our proposal, i.e., to obtain the continuous representation. Here we validate the theoretical analysis about computational complexity done in Section 4.8.1. The computational complexity is indeed quadratic with the number of centers $O(N^2)$.

6.2.2 Flux Addition

The second set of experiments consist in the study of the importance of the *free parameter* α, for the properties of the obtained solution. As explained in Section 4.5, α controls the relative importance between the terms in the functional (4.1). It means that higher values, gives a greater relevance to the *domain restrictions*, and vice versa. In these experiments we fixed the value of λ to 3. In general values of λ are in the range $[1, 10]$ show good empirical results.

In Figure 6.8 (right column) we show the results for the images in Table 6.1. We observe there is a clear decreasing tendency for the flux addition, which is what we expected by increasing the value of α. As direct consequence, there is an increase in the flux lost and the RMS of residual. This happens because the relative importance of the *similarity term* of (4.1) decreases. Also note the decreasing tendency of the sharpness, which is the response to the increasing relevance of the

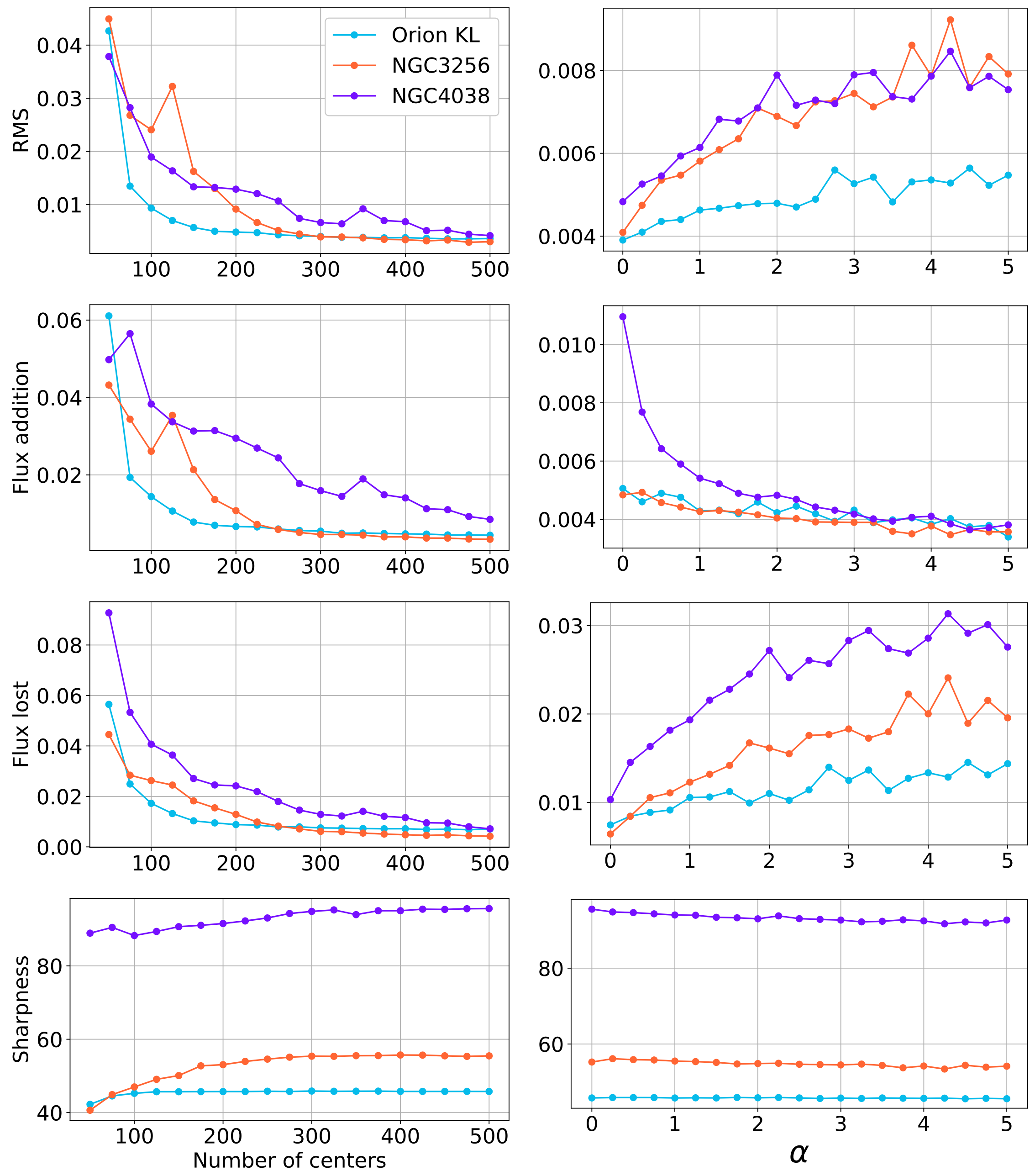


Figure 6.8: Residual RMS, Flux addition, Flux lost and Sharpness results. In the *left column* are the results for the *number of centers* experiment (with $\alpha = 0$ fixed). In the *right column* are the results for the *alpha variation* experiments (with Number of centers = 300 fixed).

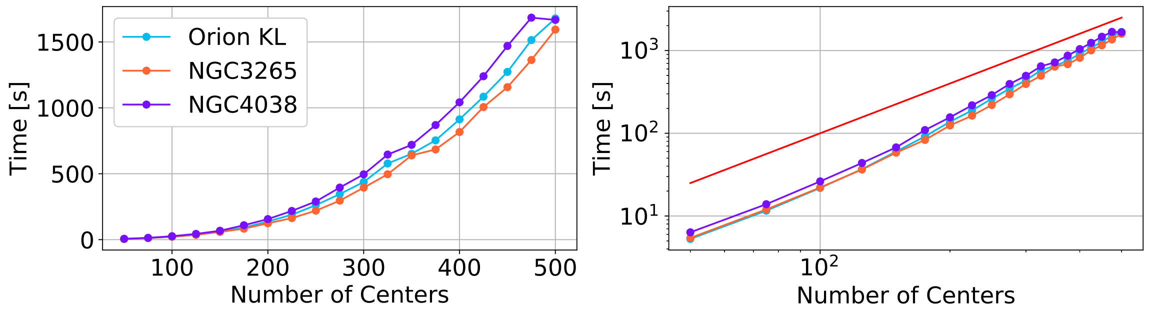


Figure 6.9: Computing times v/s the number of centers used. The red line in the right figure is a quadratic function on the number of centers, plotted to help us visualize the time complexity.

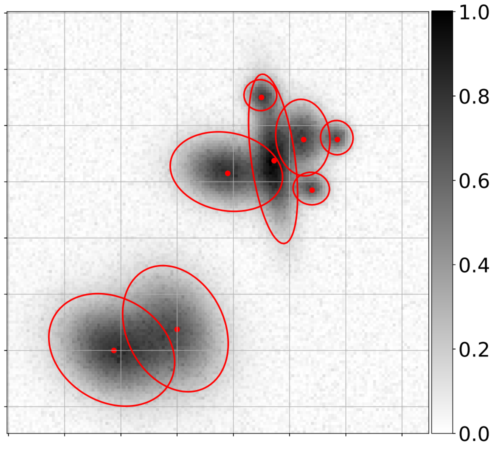


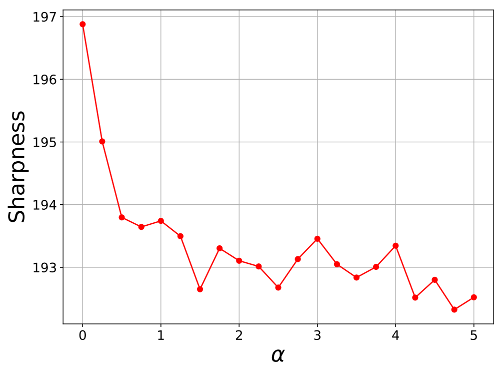
Figure 6.10: Synthetic generated data of 150×150 pixels; The red dots and ellipses indicate the positions of each Gaussian source.

domain restrictions. What happens here is that less noise peaks are being adjusted, and then the solution becomes smoother. This supports the noise reduction capacity of our proposal.

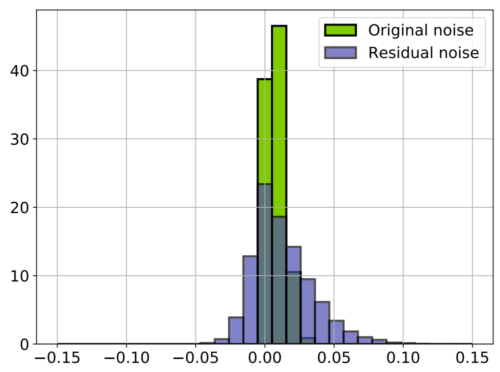
6.2.3 Noise Reduction

Here we test the noise reduction capabilities of our method over the observed data f_0. We decided to use a synthetic generated image, as shown in Figure 6.10. In this way we know exactly which part of the emission corresponds to signal and which corresponds to noise. We also know the ground truth about the clumps structure.

The image in Figure 6.10 was generated as the sum of eight blended Gaussian emissions, which represent the sources. The background additive noise was modeled with the half-normal distribution: $\epsilon \sim |\mathcal{N}(0, 0.01^2)|$.



(a) Sharpness v/s α



(b) Noise histograms. Residual noise computed with $\alpha = 2.5$.

Figure 6.11: (a) Sharpness reduction of the obtained solution u with increasing α values. (b) Comparison between the histograms of the original added noise, and the residual noise $(f_0 - u)$.

To quantitatively evaluate the noise reduction feature, we report the sharpness variation with α in Figure 6.11a. There is a clear decrease tendency in the sharpness with increasing values of α. This is the expected response because noisy images are sharper, so with higher values of α we get smoother and also less noisy solutions. We also report in Figure 6.11b the histogram of the residual noise $(f_0 - u)$ and the original noise histogram . Note that we are not retrieving exactly the noise histogram, but an approximation with a tendency toward positive values.

Two things must be mention: 1) The pixels where $u - f_0 > 0$ increment the *flux addition* measurement and correspond to the left tail of histogram of the residual noise. This could be reduce with greater values of α. 2) The pixels where $f_0 - u > 0$ increment the *flux lost* measurement and correspond to the right tail of the residual noise histogram. The tail becomes larger with greater values of α. Thus, there is a trade-off between *flux addition* and *flux lost* controlled by the parameter α.

6.2.4 Structural Analysis on Images

Finally, we show the source identification and description capabilities of our proposal. For this we apply it to real and synthetic data, and we show how the structural analysis is performed.

In Figures 6.12 and 6.13 we show three steps of the Gaussian decomposition process over the Orion-KL image, with the *reverse order* and *manual* methods respectively. We observe the *reverse order* method has problems with the leftmost source. The problem is that the method is over-decomposing it. This happens because this source could be very well represented as the combination of two or three Gaussians. Then splitting it produces a low value in the KL-bound metric. On the other hand, the *manual* method in Figure 6.13 shows very good results. We can decompose each identified source even further, but we think that the representation with 6 components it is good enough.

For comparison, in Figure 6.14 we show the results obtained by *clumpfind* and *fellwalker* algorithms. Here we display the border of the detected clumps. Similar structures are being detected for all the algorithms. However with the *manual* method of our proposal, we were able to capture and represent sources that the other algorithms couldn't find.

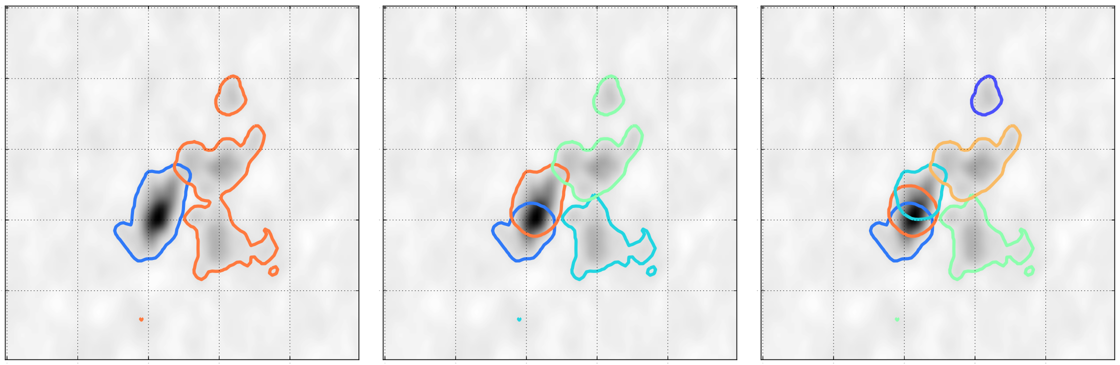


Figure 6.12: Gaussian decomposition process with **reverse order** method over Orion KL. Solution with 2, 4 and 6 components are shown from left to right.

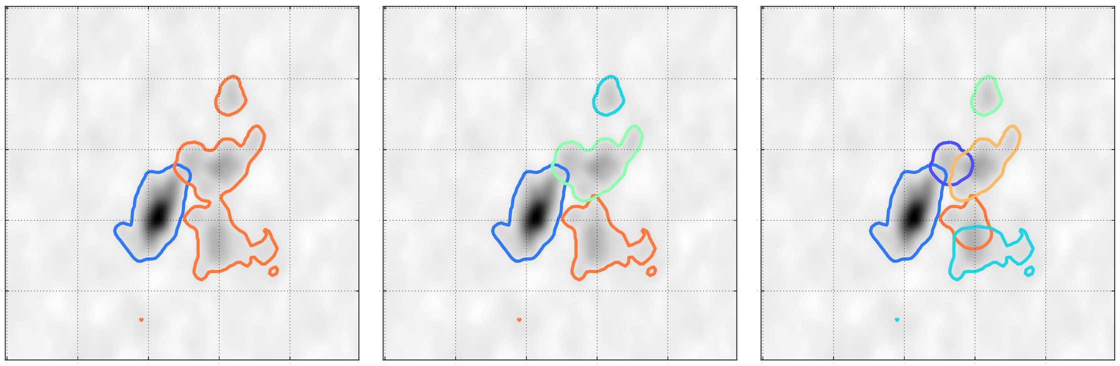
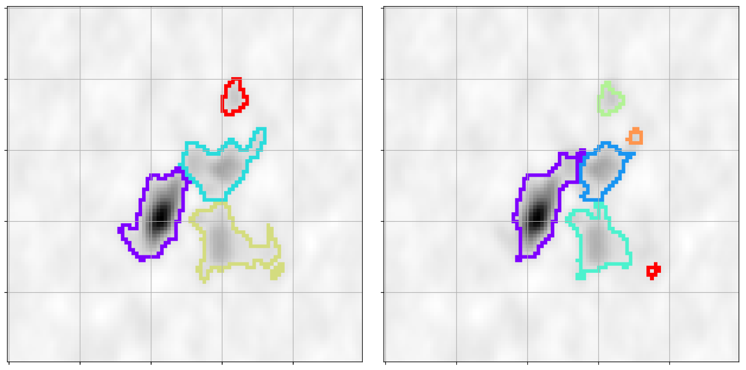


Figure 6.13: Gaussian decomposition process with **manual** method over Orion KL. Solution with 2, 4, and 6 components are shown from left to right.

Recall that our proposal does not determine the default number of clumps, as *clumpfind* or *fellwalker* do. However we offer the possibility to produce a decomposition for an arbitrary number of clumps, **with no extra significant computational cost**. This is possible since all the needed data



(a) ClumpFind **(b)** FellWalker

Figure 6.14: Segmentation results of classic clumping algorithms over Orion KL

is stored in the hierarchical tree. This is not the case of the classical clump identification algorithms, that have to be re-run with a different configuration of parameters to get a different number of clumps identified.

Due to the difficulty of validating the results (no ground-truth knowledge), we repeat the procedure in the synthetic generated image in Figure 6.10. Figures 6.15 and 6.16 show the results for the *reverse order* and *manual* decomposition methods respectively. In Figure 6.15 we see that the *reverse order* method presents the same problems as in Orion KL. It begins to decompose the bottom-left structure into more than two components. On the other side, with the *manual* method this problem is avoided (See Figure 6.16). At each step we split the clump we want to know its structure, trying to match our ground-truth knowledge about the real structure. All but one clump were correctly detected and represented. This happens because the particular clump has low intensity and spread, making it difficult to identify.

For comparison, we show in Figure 6.17 the border of the clumps detected by *clumpfind* and *fellwalker* algorithms in the synthetic image. Here we observe that *clumpfind* has serious problems with the noise, producing as result a large number of clumps identified as output. On the other hand, *fellwalker* does a very good job detecting the sources. However it is unable to capture the blended emission of coupled sources, and thus the representation of each clump is not accurate.

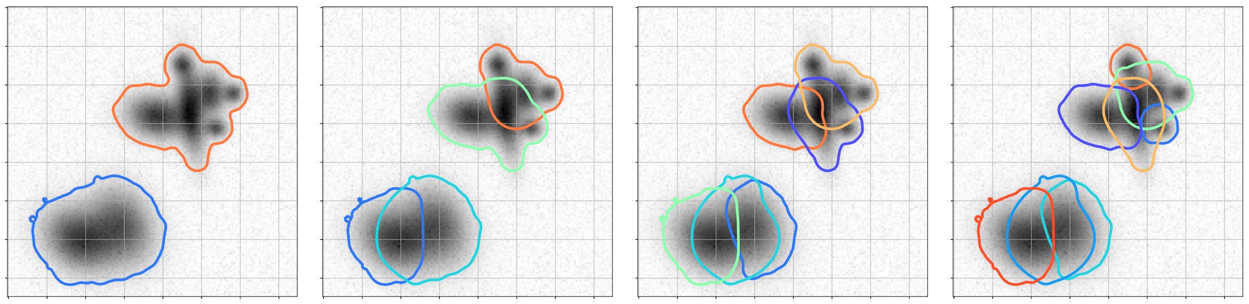


Figure 6.15: Gaussian decomposition process with **reverse order** method over synthetic data. Solution with 2, 4, 6 and 8 are shown from left to right

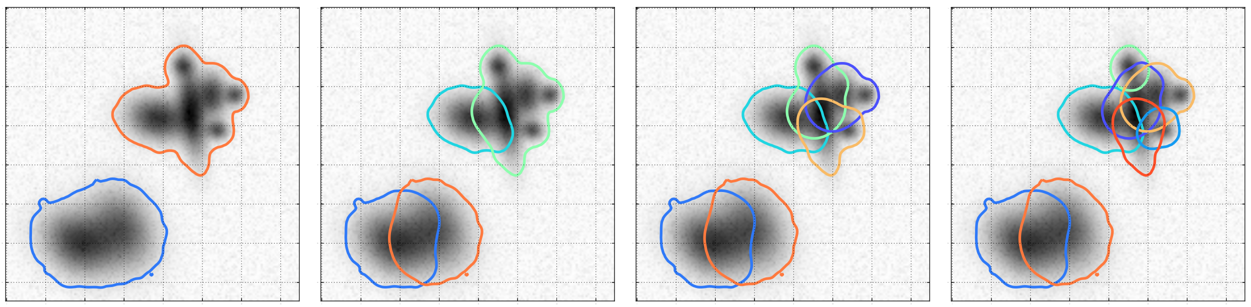
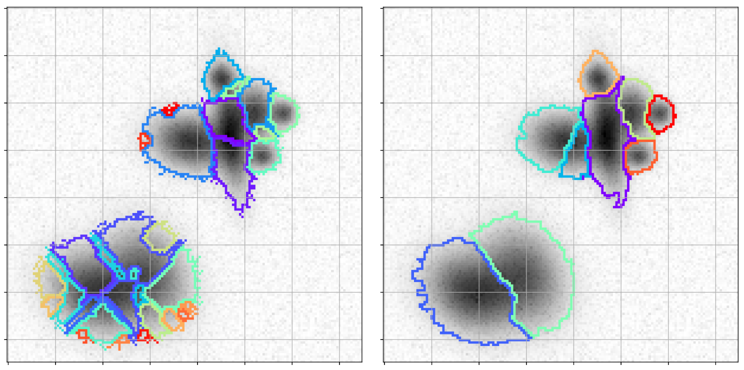


Figure 6.16: Gaussian decomposition process with **manual** method over synthetic data. Solution with 2, 4, 6 and 8 are shown from left to right



(a) ClumpFind solution. **(b)** FellWalker solution.

Figure 6.17: Segmentation results of classic clumping algorithms over synthetic data.

6.2.5 Structural Analysis on Spectroscopic Cubes

Finally we show some results in spectroscopic data cubes. We did not do all the previous experiments over these, because the current implementation is not computationally fast in 3-dimensional data. Recall that the computational time in 3D cubes increases because two reasons: the number of free parameters for each Gaussian increases (complexity $O(\gamma)$), and (most importantly) the number of Gaussians needed to get accurate representations is also higher (complexity $O(N^2)$).

In Figures 6.18 and 6.19 show final results using the manual decomposition method. These are the three 2D projections (one for each axis) of the sources identified and represented by our method. We observe in Figure 6.18 a decomposition of Orion_KL-CH3OH line cube with 5 sources. This way of visualization is useful for the manual decomposition, because each time we see a clearly disjoint structure in any of these projections, we can choose to decompose it. Figure 6.19 shows a 4-sources decomposition of HD163296-CO3_2 line cube. We observe that the decomposition is able to successfully split the clumps: Every detected clump is well differentiated from the rest in at least one projection. In this case the sources are not well distinguishable in the RA-DEC projection, but they are in the remaining two projections.

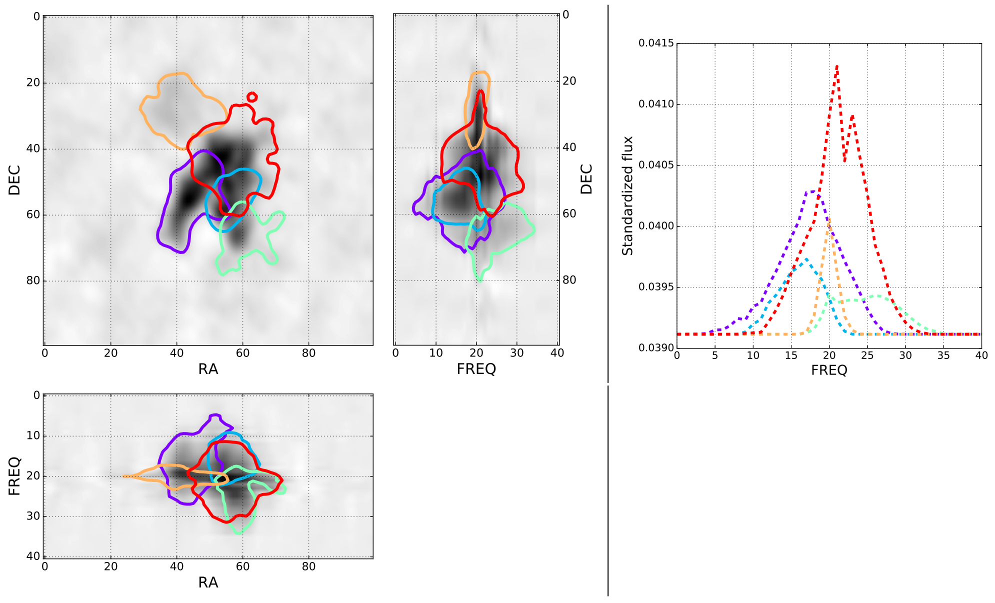


Figure 6.18: Structural Analysis over Orion_KL-CH3OH, 5 sources representation with manual decomposition. (Left) Visualization of the three 2D projections of the detected clumps. (Right) Frequency spectrum of the detected clumps, standardized by the total sum of the intensity.

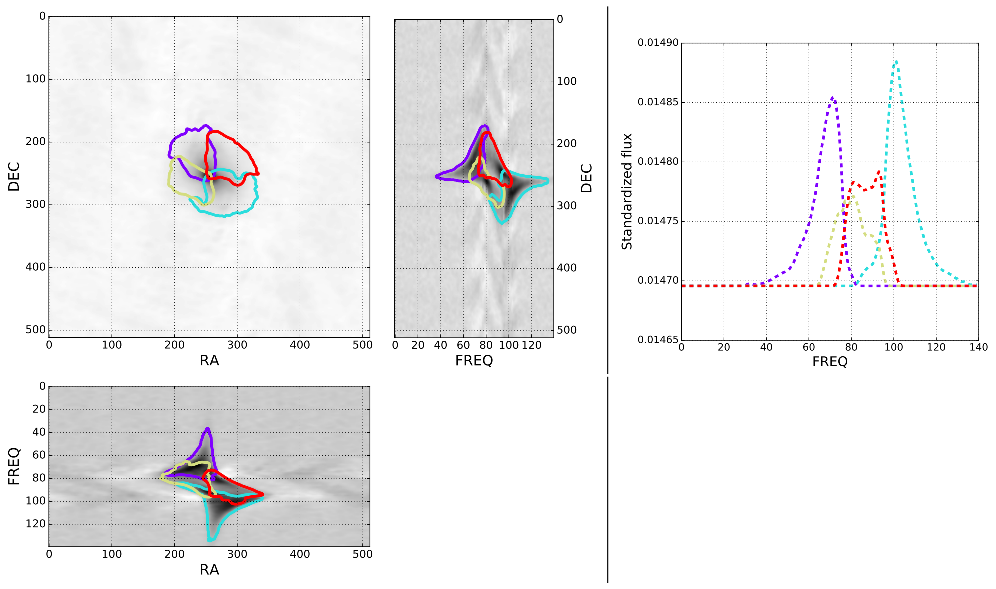


Figure 6.19: Structural Analysis over HD163296-CO3_2, 4 sources representation with manual decomposition. (Left) Visualization of the three 2D projections of the detected clumps. (Right) Frequency spectrum of the detected clumps, standardized by the total sum of the intensity.

7 | Conclusions

7.1 Conclusions

In this work we have presented a new and novel approach for structural analysis of 2- and 3-dimensional astronomical data. This was achieved in two steps: The first step produces a continuous representation of the data as a Gaussian mixture and the second step produces a hierarchical binary tree through GMR algorithms. This tree allows us to hierarchically decompose the sources or clumps identified, and perform structural/hierarchical analysis of such sources.

The contributions of this work can be separated for the two steps of the method. In the *continuous representation* step, we present a model to represent astronomical images as Gaussian mixtures. It was based on functional minimization and Levenberg-Marquardt optimization. This functional was formulated to reduce the effect of adding nonexistent flux, and to reduce the background noise. Even with the complexity of the solution space, and the bounds and simplification that we imposed, we were able to assess empirically that it produces accurate Gaussian mixture representations. In the *source identification* step, we presented a convenient alternative to the current algorithms that perform structural analysis (Alves et al., 2007; Rosolowsky et al., 2008; Men'Shchikov et al., 2012; Gregorio et al., 2014). We do not determine the number of clumps to be identified. Instead, our method produce a hierarchical tree that allows us to produce representations with any number of sources identified, with no extra significant computational cost. This is a *convenient* feature because the tree lets us manually decompose the sources we are interested in. Finally, a very important feature of our proposal is its capacity to capture the blended emission nature of the astronomical sources.

A set of experiments were performed to show the capabilities and limits of our proposal. The

Gaussian mixture representation showed to be a good alternative for the representation of sources. We also showed that as we increase the parameter α, the noise can be reduced. From the experiments over synthetic data, we observed that in most cases the residual noise is a good approximation of the original noise distribution. We were not able to retrieve the original noise distribution because the domain restrictions were imposed as soft-restrictions.

In the last set of experiments we presented qualitative results for the structural analysis approach of our work. We demonstrated how automatic and manual decomposition over 2D images and 3D spectroscopic cubes work. The automatic strategy gave good preliminary results but there are space for improvements. The automatic strategy wasn't *robust* enough, and did some mistakes by over-decomposing the sources. However it opens a challenge: Create a measure or metric to perform a *robust* automatic decomposition of the sources. On the other hand, the manual decomposition strategy proved to be *convenient* and a useful tool to perform structural analysis. This because it allows the expert to know the hierarchical structure of the source of interest, depending of the science case of study.

We think our work contributes to two unexplored areas of this problem: 1) Producing a continuous representation of the data, and performing the identification of clumps over this representation. This allows us to capture the blended nature of the emissions. 2) Creating a hierarchical tree data structure allows the user (astronomer) to decompose the sources arbitrarily with no significant extra computational cost.

We have successfully proposed, analyzed and compared a novel algorithm for hierarchical decomposition, identification and quantification of sources over blended astronomical data. It has shown to be a convenient alternative to state-of-art algorithms.

7.2 Future Work

There are many improvements and research directions that we would like to explore in the near future. We present a list of them:

1. Right now the major computational *bottleneck* is the evaluation of the residual function (4.10). In order to apply our method on cubes of large dimensions, we need faster evaluation of this function. This could be achieved using domain decomposition methods and sparse

representations, for example.

2. It would be interesting to see if there are significant improvements in the obtained representation, by allowing the Gaussians to be elliptically symmetrical.

3. Implement and test the more advanced GMR algorithms. This is to verify if we get a more robust hierarchical tree. Also, it would be great to estimate the *minimum* number of components for which the mixture does not degenerate.

4. Linked to the previous point, it would be interesting to study automatic strategies for the decomposition of the hierarchical tree.

5. The two steps of the method are totally independent, i.e., we can use any representation of the image as Gaussian mixture and perform GMR over this. We propose this method for the Gaussian mixture, because allows us to handle the no flux addition and noise reduction restrictions. It would be interesting to try different and simpler approaches, that do not require nonlinear optimization.

6. Although the process of obtaining the Gaussian mixture through least squares minimization works empirically well, we cannot ensure an optimal solution is being reached. It would be interesting to search different Gaussian models, that ensure us that we are obtaining an optimal solution.

Bibliography

Alves, J., Lombardi, M., Lada, C., 2007. The mass function of dense molecular cores and the origin of the IMF. Astronomy & Astrophysics 462 (1), L17–L21.

Araya, M., Mendoza, M., Solar, M., Mardones, M., Bayo, A., 2017. Analysis of N-dimensional Astronomical Images Through Homogeneous Representations, manuscript in revision for publication in Astronomy and Computing.

Berry, D., 2015. FellWalker—A clump identification algorithm. Astronomy and Computing 10, 22–31.

Berry, D., Reinhold, K., Jenness, T., Economou, F., 2007. CUPID: a clump identification and analysis package. In: Astronomical Data Analysis Software and Systems XVI. Vol. 376. p. 425.

Bertin, E., Arnouts, S., 1996. SExtractor: Software for source extraction. Astronomy and Astrophysics Supplement Series 117 (2), 393–404.

Chen, H., Chang, K., Smith, C., 2010. Constraint optimized weight adaptation for gaussian mixture reduction. In: SPIE Defense, Security, and Sensing. International Society for Optics and Photonics, pp. 76970N–76970N.

Crouse, D. F., Willett, P., Pattipati, K., Svensson, L., July 2011. A look at Gaussian mixture reduction algorithms. In: 14th International Conference on Information Fusion. pp. 1–8.

Faure, H., Lemieux, C., 2009. Generalized Halton sequences in 2008: A comparative study. ACM Transactions on Modeling and Computer Simulation (TOMACS) 19 (4), 15.

Gregorio, R., Solar, M., Mardones, D., Pichara, K., Parada, V., Contreras, R., 2014. Automatic detection and automatic classification of structures in astronomical images. In: SPIE Astronomical Telescopes+ Instrumentation. International Society for Optics and Photonics, pp. 91522R–91522R.

Gupta, A., 1996. Calculus of variations with applications. PHI Learning Pvt. Ltd.

Hills, R., Jan 2011. ALMA Science Verification. ALMA Newsletter, 7,4-7.

James, F., Roos, M., 1975. Minuit-a system for function minimization and analysis of the parameter errors and correlations. Computer Physics Communications 10 (6), 343–367.

Kennicutt Jr, R. C., Evans, N. J., 2012. Star formation in the Milky Way and nearby galaxies. Annual Review of Astronomy and Astrophysics 50, 531–608.

Logan, J. D., 2013. Applied mathematics. John Wiley & Sons.

Men'Shchikov, A., André, P., Didelon, P., Motte, F., Hennemann, M., Schneider, N., 2012. A multi-scale, multi-wavelength source extraction method: getsources. Astronomy & Astrophysics 542, A81.

Moré, J. J., 1978. The Levenberg-Marquardt algorithm: implementation and theory. In: Numerical analysis. Springer, pp. 105–116.

Rosolowsky, E., Pineda, J., Kauffmann, J., Goodman, A., 2008. Structural analysis of molecular clouds: Dendrograms. The Astrophysical Journal 679 (2), 1338.

Runnalls, A. R., 2007. Kullback-Leibler approach to Gaussian mixture reduction. IEEE Transactions on Aerospace and Electronic Systems 43 (3).

Schieferdecker, D., Huber, M. F., 2009. Gaussian mixture reduction via clustering. In: Information Fusion, 2009. FUSION'09. 12th International Conference on. IEEE, pp. 1536–1543.

Stutzki, J., Guesten, R., 1990. High spatial resolution isotopic CO and CS observations of M17 SW-The clumpy structure of the molecular cloud core. The Astrophysical Journal 356, 513–533.

Villanueva, M., Araya, M., 2017. Multiresolution and Hierarchical Analysis of Astronomical Spectroscopic Cubes using 3D Discrete Wavelet Transform, To be published in proceedings of the ICPRS-2017.

Wang, X., Serpedin, E., Qaraqe, K., 2014. Variational Methods in Signal and Image Processing. In: Advances in Mathematical Models and Production Systems in Engineering. WSEAS, pp. 11–16.

Watson, M. E., 2010. Assessing the Performance of Sub-Millimetre Compact Object Detection Algorithms. Master's thesis, University of Hertfordshire.

Williams, J. L., Maybeck, P. S., 2003. Cost-function-based Gaussian mixture reduction for target tracking. In: Proceedings of the sixth international conference of Information fusion. Vol. 2. pp. 1047–1054.

Williams, J. P., De Geus, E. J., Blitz, L., 1994. Determining structure in molecular clouds. The Astrophysical Journal 428, 693–712.