KØBENHAVNS UNIVERSITET

MSc Thesis

Thomas Ravinet

AutoMUSE : Autonomous MUSE Source Extractor

Automatic detection and extraction of MUSE spectra

Directed by Lise Christensen and Adriano Agnello

NBI/DARK/May 31, 2021

Abstract

The quantity and quality of data within astronomical archives are growing. With large-scale surveys of the night sky, new tools must be developed in order to process an unprecedented amount of information, in a standardized way. Additionally, recent tools, called **I**ntegral **F**ield **S**pectrographs, gather spectra from an entire field of the sky. Searching for astronomical sources in the 3-dimensional (Right Ascension, Declination, and wavelength) images they produce can be difficult. Using **IFS**, 3D pictures of specific objects can be taken, without analyzing other sources in the image, such as in the background or on the edges.

In this Master of Sciences thesis report, we propose to create a tool, **AutoMUSE**, standing for **Auto**nomous **MU**SE **S**ource **E**xtractor. AutoMUSE is dedicated to the analysis of MUSE instrument products, but should in principle work on other IFS datacubes. The tool is compliant with (most of) the FITS guidelines, and produces spectra, catalogs and images in a standardized way.

After a description of the source images we are interested in, we will describe the functioning of our software, and of the mechanisms behind it. We apply it to three different fields, and show results form that analysis. Additionally, we worked on attempts of automatic classification of spectra, and describe our reasoning.

Some definitions

In the report, we will use the following terms:

Datacube	3D astronomical image, with two spatial — generally right ascension and declination — and one spectral — generally wavelength — axes, containing **voxels**. Each **voxel** contains a value, generally being a flux at a certain wavelength.
Voxel	Pixel, generalized to a 3D space.
Image	2D astronomical object with two spatial axes. An image of 5 pixels — or voxels — along the spectral axis from the datacube is the projection (through sum, average...) along to spatial axes of a 5 spectral pixels selected region of the datacube.
Stat (extension)	Extension containing data uncertainty on the source datacube/image. Stat is composed of 2D pixels or 3D voxels. Can be represented as a standard deviation σ, a variance σ^2 or a weight $\frac{1}{\sigma^2}$.
Broad-band image	An image from a large spectral portion of the datacube, projected to spatial axes. White light is the image of the whole cube, blue the first

third of the wavelength found in the datacube, green of the second and red of the last.

Narrow-band image An image of a few wavelengths of the datacube, projected to spatial axes.

Segmentation map : an image representing position and extent of various objects present in an image or a datacube. To each object is assigned a unique integer.

Acknowledgments

First, thanks a lot to Lise and Adriano for helping and accompanying me along this thesis! It was a fascinating job and I learned a lot during it. Thanks a lot to Hannah for helping me with figures of this thesis report, to Valerie who checked spelling errors. Also, thanks to Essenam for letting me speaking astronomy for hours.

Also, anticipated thanks to Jérôme for censoring this thesis!

Finally, mandatory acknowledgments :

Thanks to SCIENCE HPC Center and ERDA for letting me use their facilities, without which I would not have been able to run and test the majority of my AutoMUSE analysis.

Thanks to SDSS, as I used their data: Funding for the Sloan Digital Sky Survey IV has been provided by the Alfred P. Sloan Foundation, the U.S. Department of Energy Office of Science, and the Participating Institutions.

SDSS-IV acknowledges support and resources from the Center for High Performance Computing at the University of Utah. The SDSS website is www.sdss.org.

SDSS-IV is managed by the Astrophysical Research Consortium for the Participating Institutions of the SDSS Collaboration including the Brazilian Participation Group, the Carnegie Institution for Science, Carnegie Mellon University, Center for Astrophysics | Harvard & Smithsonian, the Chilean Participation Group, the French Participation Group, Instituto de Astrofísica de Canarias, The Johns Hopkins University, Kavli Institute for the Physics and Mathematics of the Universe (IPMU) / University of Tokyo, the Korean Participation Group, Lawrence Berkeley National Laboratory, Leibniz Institut für Astrophysik Potsdam (AIP), Max-Planck-Institut für Astronomie (MPIA Heidelberg), Max-Planck-Institut für Astrophysik (MPA Garching), Max-Planck-Institut für Extraterrestrische Physik (MPE), National Astronomical Observatories of China, New Mexico State University, New York University, University of Notre Dame, Observatário Nacional / MCTI, The Ohio State University, Pennsylvania State University, Shanghai Astronomical Observatory, United Kingdom Participation Group, Universidad Nacional Autónoma de México, University of Arizona, University of Colorado Boulder, University of Oxford, University of Portsmouth, University of Utah, University of Virginia, University of Washington, University of Wisconsin, Vanderbilt University, and Yale University.

Contents

Chapter 1

Introduction

Along this thesis, we are going to derive a way to detect and extract objects in peculiar three-dimensional images, and attempt to classify them. Before describing existing processes and explaining how the process we implemented works, in chapters 3 and 4, and presenting results and possible improvements, in latter chapters, we will start by an introduction about astronomical concepts, in chapter 1. In chapter 2, we will discuss instruments and data structure of the tools and files we use in AutoMUSE.

1.1 Night sky and collection of light

While looking at the night sky with naked eye, one can see bright point-sources, stars, and more nebulous and extended objects, such as the Milky Way, nebulae, and nearby galaxies. Some faint ones can be observed without any instrument if there is a great, dry sky, with no light pollution around — the last point being a growing problem as stated in [16] [22]. However, even with excellent observing conditions, an astronomer will not be able to distinguish the billion stars compounding the Milky Way with its sole eyes, nor to identify colors of distant objects. Thus, humanity and scientific community had to come with ways to collect more light.

In order to achieve this problem, telescopes were invented. A telescope is a device composed of lenses and mirrors (for their simplest forms), and of an eyepiece allowing an observer to look at the celestial archway. The use of a telescope greatly increases the surface collecting light: while a pupil is $\approx 6 - 7$mm diameter wide, a telescope can have much larger surfaces: commercial telescopes for amateurs easily reach an aperture of 250 to 300mm, while professional optical telescopes diameters can be several meters. The emblematic Hubble Space Telescope has a 2.4m mirror, and ground telescopes light-gathering pieces can reach 10.4m — the aperture size of the Gran Telescopio Canarias.

With telescopes, not only astronomers observe fainter, nearly invisible objects, but also increase the resolution of observations, their ability to distinguish close objects in the sky.

We use the term "resolution" to mark how two objects close in the sky appear as merged while looking at them with a telescope. It is defined as the ratio between observed wavelength and telescope diameter; in the case of visible light, the resolution power is often taken at 550nm.

Using the resolution, one can define "criterions", such as the Rayleigh criterion, that permits qualifying if two objects, angularly close in the sky, can be deblended:

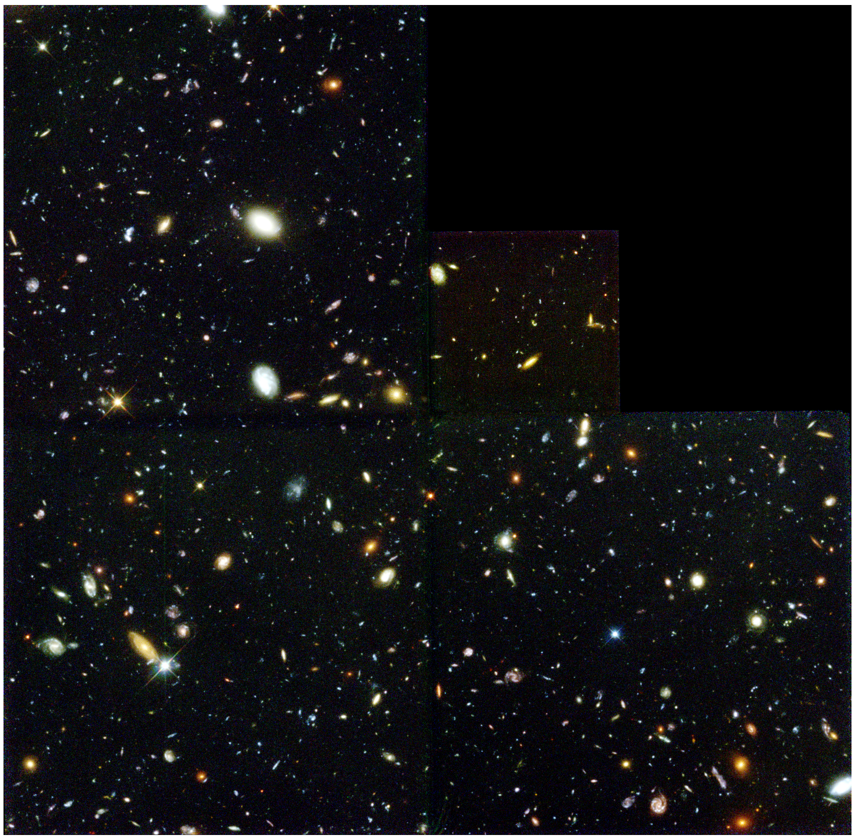

Figure 1.1: The Hubble Deep Field, R. Williams (STScI), the Hubble Deep Field Team and NASA.

$$\sin \alpha_R = 1.22R = 1.22\frac{\lambda}{D}$$

Using that criterion, one can find the minimal angular size α_R of an object to be resolved with a particular telescope. If we manage to know the distance from an observed object, we can also be able to determine its physical size. For example, we can show that in the visible light, the minimal detail size of an object on the Moon (distant of roughly 384.000km) with naked eye is about a hundred kilometers, while the Extremely Large Telescope (E-ELT) will distinguish 6-meters wide objects with its 39m aperture.

However, increasing the size of the mirror used in a telescope is not the only way to collect more light. With the invention of photographic plates, and after that of more sophisticated light collectors, such as CCDs or CMOS sensors, astronomers can register light/photons from a source for a longer time, thus gaining the ability of observing the faintest objects of the universe. Hubble Space Telescope revealed that a 24-millionth of the sky, while observed for $\approx$ 35h in total, at well-chosen wavelengths, contains a thousand of galaxies.

Thus, we have two ways to increase the observing power of a telescope. Both comes with sources of error, such as chromatic aberration or cosmic ray impacts.

1.2 Analysis of the night sky

What can we learn from images such as the Hubble Deep Field (Figure 1.1)?

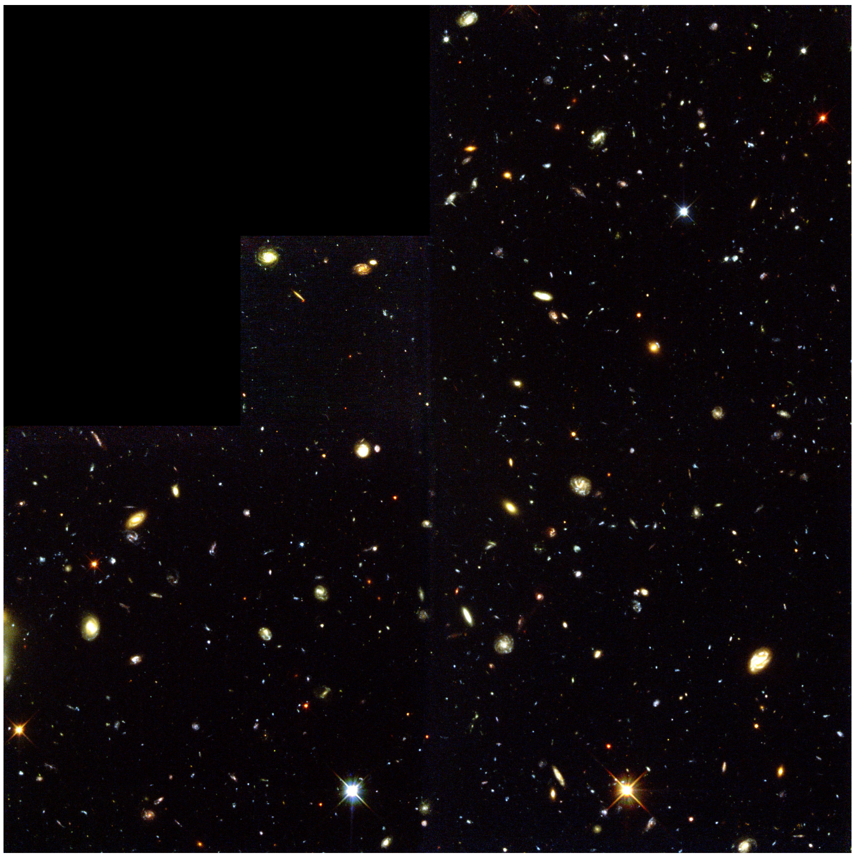

Figure 1.2: R. Williams (STScI), the HDF-S Team, and NASA/ESA.
Hubble Deep Field - South looks similar to the North one 1.1. Hubble Deep Fields demonstrated that observing dark, small places of the sky reveal old galaxies, with different shapes than the nearby observed ones. It also, with different pointings, is an illustration, if not a proof, of the cosmological principle.

The first element concerns the nature of observed objects. Being able to build telescopes allows, as we stated, to increase the resolution of our imaging of the night sky. Forbes [19] , already in 1909, tightly linked the technical progress to the evolution of our understanding of the Universe.

Historically, scientists and philosophers were first able to make the difference between stars, that look like point sources, and nebulae, where all extended objects were classified, without understanding their nature and dynamics. With technical progress, the difference between galaxies and dust clouds was demonstrated by Hubble. The famous astronomer showed that the fuzzy aspect of galaxies was due to a small telescope resolution, that was not sufficient to make the distinction of billions stars laying outside our own Milky Way. The other dust clouds can be linked to other phenomena, such as supernova events.

With the increasing quality of observations, a giant leap in understanding the Universe's laws occurred during the XX$^{\text{th}}$ century. General relativity unveiled relations between space and time, that observing furthest objects permits unrolling history to the birth of our Universe. In recent years, space borne observatories demonstrated that dark, small pieces of the sky are full of information; mapping them could confirm and refine equations ruling the evolution of the cosmos.

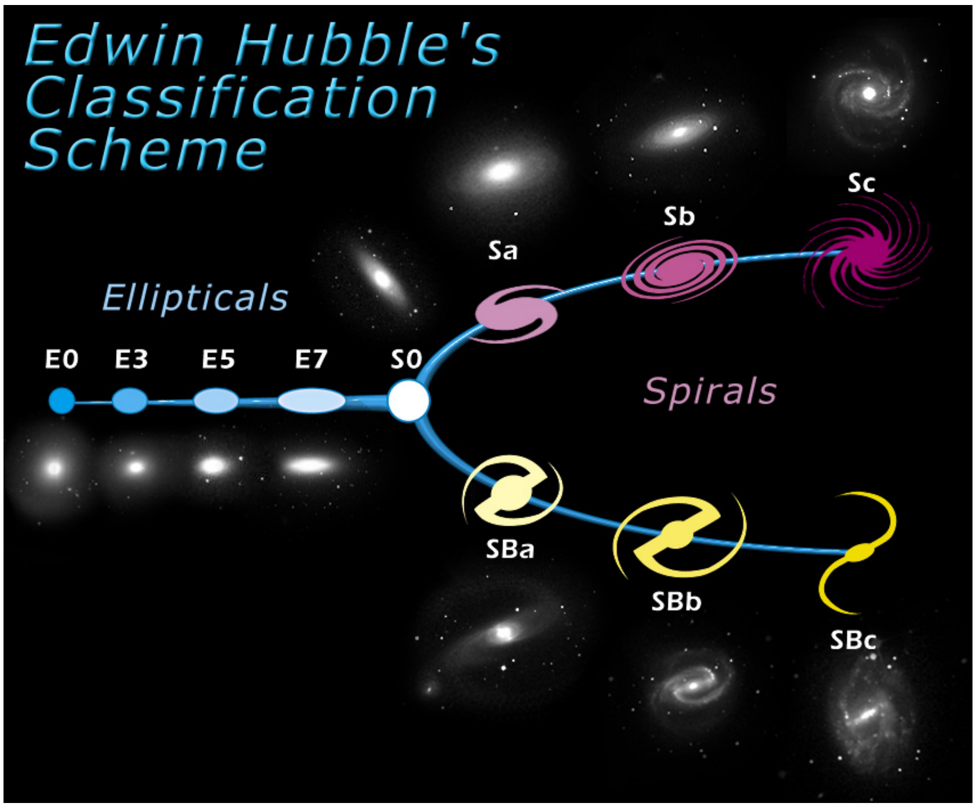

Figure 1.3: Hubble's classification scheme, Wikipedia Commons.

1.2.1 Shape of galaxies

Mapping the sky requires observation facilities, that point to different regions and objects around the celestial sphere. Once these observations are done, they have to be understood and analyzed.

A first analysis method is using simple imaging telescopes and sensors, to attempt classifying the celestial objects based on their shapes and colors.

One could separate stars, looking like point sources, from fuzzy objects fixed in the night sky, and from objects that appear to be moving in the sky, such as comets and planets. The latter celestial corpses belong to our local galaxy, or even to our solar system, and can be characterized through their luminosity. However, one would need additional observations (see section 1.2.1) to understand their nature, similarly to stars.

More fuzzy objects were studied by Edwin Hubble in the 1920s'. Observing more than 400 "extra-galactic nebulae", in order to determine their nature, the astronomer produced a classification system to categorize his findings, Figure 1.3.

The Hubble Fork was first described by Hubble in its 1926 article,[27]. As Galaxy Zoo stated in a blog post, it is still one of the golden standards in morphological classification of galaxies. The fork in Figure 1.3 distinguish elliptical from spiral and barred spiral galaxies. Each galaxy that light distribution seems elliptical is associated to an integer, ten time the ellipticity of the object (rounded). For example, an E7 galaxy is elliptical, with its ellipticity being $e = 1 - \frac{b}{a} \approx 0.7$ (a and b being the semi-major and semi-minor axes of the light distribution in the galaxy picture).

Spiral galaxies are denoted S, and can comport a star "bar" through it, that will be denoted SB. The fuzziness and tightness to the central bulge of galaxies' arms are rated from "a" (diffuse and tight to the bright core arms) to "c" (extended, resolved arms around a fainter core).

However, the Hubble classification was not complete, and did not account for some galaxies particularities. In his Galaxy Morphology article [14], Buta traces the various changes made to the Hubble galaxy classification system. He denotes that Sandage added the S0 type, representing galaxies with a central bulge, as well as a disk around it, but no spiral arms. Sandage also carried two sub-types, allowing to take into account if a galaxy contains a “ring” (r) in its light distribution, or if it’s a pure spiral (s).

In its own classification system, De Vaucouleurs enriched the Hubble-Sandage classification, into what he referred as a volume. The first dimension would be the *stage* of the galaxy, being the axis going from elliptical to spiral galaxies in the Hubble Tuning Fork. The two other axes represent a central bar presence or absence, as well as the presence of a ring. Sandage - de Vaucouleurs also enriched the definition of classifications of irregular galaxies, that cannot enter in any cases of the various tuning forks.

De Vaucouleurs also pointed out that another classification can be done comparing esteemed mass - or diameter - to colors of galaxies in [17].

As Baldry points in [6], the word stage, used by Hubble and frequently by astronomers, has no correlation to time evolution of the Universe. The “early” type galaxies, on the left of the tuning fork, are not to be taken as if all the first-generation galaxies were elliptical.

The morphological classification has limits: galaxies evolve in a three-dimensional space (plus time) projected on the two-dimensional sky surface. Thus, all galaxies are not face-on, and the observed ellipticity is the projected ellipticity. Additionally, the morphological classification system, in the visible light, is barely linked to any physical parameter, as [14] reminds us, particularly for elliptical galaxies [31].

Finally, cataloging galaxies morphology is an important work in the framework of large-scale survey. Studying distributions of shapes improves our understanding of celestial objects formation. The Galaxy Zoo project is a collaborative tool aimed at classifying galaxies in order to draw statistics and physics from galaxy shapes, such as the star formation rate of spiral galaxies [53], or to identify peculiarities in shapes, informing us about formation events - such as mergers. However, galaxy shape and luminosity are not sufficient to characterize it. The intensity and luminosity profile of objects are not the only source of data.

Spectra and redshift

During the XVII$^{\text{th}}$ century, Newton discovered that projecting a sunlight ray in a prism would decompose it in a rainbow. That crucial discovery led to understand that light is an (electromagnetic) wave.

Nevertheless, light sometimes also behaves like a collection of particles, *photons*, containing a specific level of energy, a *quantum*. This level of energy can be quantified by a number, λ, the wavelength.

The paradox that light can be seen as both wave and particles has useful consequences. For example, one can collect light, and count how many photons of specific energy - that is, wavelength - are composing the wave. This count-per-wavelength is called a spectrum.

Before understanding light nature, we used our (human) eyes to observe the sky. Eyes are adapted to see light with wavelengths ranging from 380 to 700nm, through the optical spectrum. With the discovery that there is light outside this range, astronomers built instruments permitting to watch what happens beyond visible spectrum. Then, they could link photons distributions along wavelength to different physical processes.

Observers can look at celestial bodies, and using a spectrograph, decompose their light into spectra.

While looking at spectra of stars in the optical range (300 − 800nm), one can find several features and light behaviors, as on Figure 1.4.

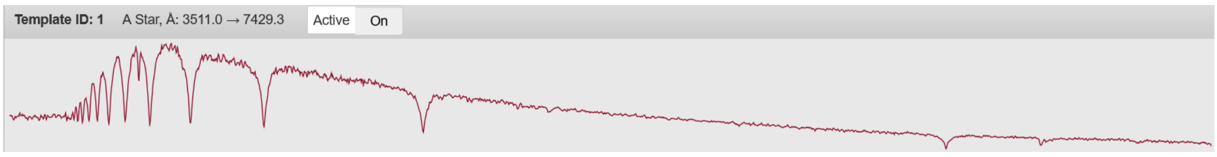

Figure 1.4: A-type star template from Marz software [24].

These features can be explained by exposing observations to the light of quantum physics.

A first element that gives a star spectrum its shape is its continuum. If the dips in the spectrum are ignored, and that the noise due to observing conditions is removed, a spectrum can be smoothed to a continuous and slowly varying function depending on wavelength, the continuum. It can be related to the Planck's law :

$$B(\lambda, T) = \frac{2hc^2}{\lambda^5} \frac{1}{e^{\frac{hc}{\lambda k_B T}} - 1}$$

This law links the spectral radiance of a body to its temperature. It is also called a black-body spectrum. Using it, we can, for example, determine the temperature of a stellar body.

We also notice, on the spectra, "dips", wavelength ranges where the stellar object emits less than the continuum. These features are explained by quantum mechanics and atoms behavior. A star's atmosphere is composed of atoms and ions. Electrons, fundamental compounds of these, can absorb specific and discrete quanta of energy, that are only dependent to the nature their associated element. Due to that quantization, an electron can be excited to specific levels of energy, jumping discontinuously from one to another. At the surface of a star, the energy that can excite an electron to a level higher than its ground, fundamental state, is carried by photons, that have an energy depending on their wavelength: $E = \frac{hc}{\lambda}$.

When a photon, emitted by the black-body - thermal - radiation of the star, having a specific energy is absorbed by an electron at the right state, it creates an absorption feature in the spectra. The "intensity" of these features will be dependent to the concentration in a star atmosphere of the atoms and ions, and of their attached electrons.

As we can measure (and compute), on Earth, which atoms and ions can absorb photons at a specific wavelength, it is possible to trace back stellar atmosphere composition and behavior from absorption lines.

When looking at a galaxy spectrum, we notice a different pattern.

The various galaxies optical spectra in figure 1.5 have other kind of features. Their continuum cannot be fitted with a Planck function, but its flux seems to be increasing to a plateau starting at ≈ 4000Å - for the two first ones. They also have both absorption and emission lines.

Nevertheless, all these features can be explained by the nature of galaxies: such an object is the sum of stars and gas.

As explained by Buat in a lecture [13], the continuum emission of the galaxies in the optical range is due to the sum of light emitted by the stars.

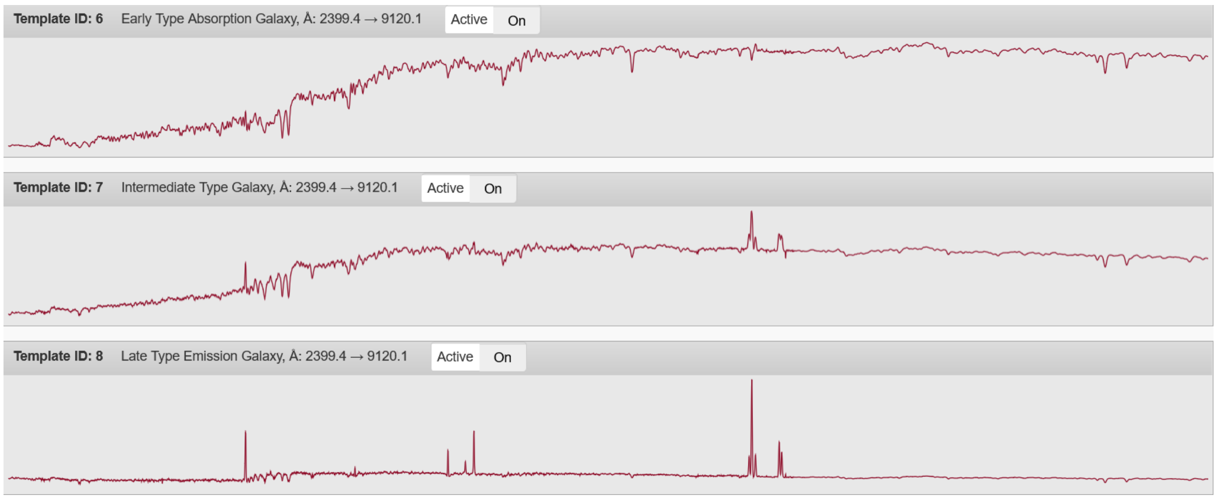

Figure 1.5: Optical galaxy spectra templates, from Marz software [24].

The hot, new stars in the nebulous object heat-up the gas surrounding them, thermally exciting it. The atoms and ions compounding that gas will de-excite by emitting photons at allowed frequency, thus making appear emission features.

Finally, the absorption lines in the galaxy characterize an old population of stars, whose atmosphere's metals absorbs the light at key wavelength, with the same mechanism as stated above.

When observing spectra, the lines can be narrow or broad, and the whole spectra can be shifted to blue or red. This is due to Doppler effect, that affects the perceived frequency/wavelength of the observer relatively to the (radial) velocity differential between source and observer. Broad absorption line in a star spectrum can translate the thermal agitation of the absorbing atmosphere, while a rotating galaxy increases the width of its emissions lines. If the entire line is moved, it means that the object moves relatively to us, as characterized by the redshift formula (in the non-relativistic case):

$$z = \frac{v_{||}}{c}$$

where

$$z = \frac{\lambda_{obs}}{\lambda_{emit}} - 1$$

Hubble noticed that the galaxies velocities, when "sufficiently far away", follow the formula

$$v = H_0 D$$

, where $H_0 \approxeq 70$km/s/Mpc and D the distance to the galaxy. By showing that empirical law, the astronomer demonstrated that the Universe is in expansion.

1.2.2 Why should we map the Universe?

Mapping the universe is not only about being able to place objects on the sky — even if this has been quite useful for navigation before the invention of GPS — but also about characterizing them. Thanks to general relativity, being able to classify objects redshift grant access to their ages as well as their nature.

A classification of objects across redshift, even if done roughly, can be used to derive elements formation history, or star formation rate evolution in galaxies. It would give access to the great story of how galaxies evolve across millions (billions) of years, and the evolution of cosmological dynamics and interactions.

A shape-or-color based classification, that would not require spectrographs, would also be useful. It could bring information on the relative populations of celestial bodies. It would also be less expensive than wide spectrometric studies. Using knowledge brought by spectral classification, one can downgrade data quality to colors in broad-band images, simulating the usage of broad-band instruments. From data acquired by spectral analysis, one can then derive properties from objects using only flux, intensity of objects, in various broad-band images.

In this thesis, we focus on developing a method to provide information on the location of faint, distant objects, automatically, and incurs in the territory of automatic classification, using the capacities of a recent (2014) instrument, MUSE.

Chapter 2

Related works

In this thesis, we are searching for astronomical sources on images, captured by MUSE instrument. Such a work is called "construction of a catalog".

2.1 Astronomical catalogs

Astronomical catalogs are a way of listing the different celestial bodies observed, by various techniques and instruments, as well as information derived from these observations. Catalogs usually contains, for each object, a name, coordinates, and some information about what is observed — it can be photometry, analysis, or an eye-description of the object. We can also find information about redshift of a galaxy, or metallicity of a star, if it was the topic of the catalog built. Forbes [19] assumes that the firsts catalogs were constructed by Chinese civilizations, but also describes how careful Arabic scientists, during the first millennia, gathered with accuracy and method object coordinates in the sky. Western astronomers later produced tables of celestial bodies, similarly to current methods. Charles Messier, a French comet hunter from the XVIth century, discovered and listed deep-space objects in its Messier Catalogs. Another notable catalog is the New General Catalog (NGC), containing thousands of objects, and that was corrected and revised until 2015[1]. The catalogs are now produced by surveys, instruments or method of discovery. This led to an increase of numbers of catalogs, covering different portions of the sky, in various wavelength ranges and deepness. To mine through catalogs, tools, such as Vizier[2], exists, permitting to identify objects in various catalogs through names, coordinates, and numerous other parameters.

The tool also includes cross-identification attempts of objects. However, the matches must be taken carefully, for multiple reasons: an instrument taking images may have an offset on coordinates, the object could be a false detection, or two objects can lay really close to each other, or even in the same line-of-sight.

2.2 Instruments used for Spectroscopic Astronomical Surveys

In this thesis, we are interested into producing automatically catalogs from MUSE patches of the sky, that may contain either stars and galaxies, and to extract their associated

[1]`http://ngcicproject.observers.org/`
[2]`https://vizier.u-strasbg.fr/vizier/`

spectra, profiting from MUSE capacities- see subsection 2.2.2.

Multiple catalogs and surveys exist to describe portions of the sky. However, few instruments were created with the ability to gather images and spectra during the same time.

2.2.1 Sloan Digital Sky Surveys - SDSS

Sloan Digital Sky Surveys are long term projects to map the sky, and to obtain science from its maps.

SDSS projects occurred in four phases, with a fifth one starting in 2020/2021. Between 1998 and 2008, a 126 megapixels' camera observed 30% of the sky, in 5 different color bands[3].

While finding objects on the band-images, some of them were (and still are) selected in order to obtain their spectra. To capture these, drilled-aluminum plates were placed in the telescope, with holes at the positions of celestial bodies to analyze. Optical fiber linked target to a spectrometer, that deliver spectrometry for up to a thousand objects per plate.

During the first phase of SDSS, around 800 000 galaxies and 100 000 quasars were mapped with spectra. In the later phases, the telescope also observed spectroscopically more than 300 000 stars (through SEGUE and SEGUE 2 experiments) , analyzed spatially distributed spectra of nearby galaxies (MaNGa). The BOSS and eBOSS missions found the redshift for more than a million objects.

The SDSS data releases are key instruments for astronomical research. Freely accessible spectral and photometric databases allows performing analysis on numerous topics. M. Tegmark and al. [47] used SDSS to refine cosmological parameters, while the SEGUE missions [52] permitted to explore abundances of elements in Milky Way 's stars and to explore chemical evolution of its disk [43], among other discoveries.

Another key point of SDSS surveys is its identification pipelines, permitting to classify automatically (numerous) gathered spectra. `redmonster` and `idlspec2d` are template-matching tools, reliable more than 95% of time [11], used to automatically match spectra. The SEGUE pipeline[4] permits estimating stars parameters, when a good signal-to-noise ratio is detected.

Finally, the use of massive data set of spectra from SDSS repository can be used to train regression machine learning algorithms for redshift and categorization [26]. Indeed, we will use SDSS spectra for our automatic classification of spectra algorithm in subsection 6.1.3

The limits of SDSS studies is the rate at which it can acquire spectra: holes must be drilled on a cartridge, then each hole of the plate is manually plugged with optical fiber to spectrometer, before an observation can occur. The SDSS-V will fix that spectra acquiring system by using a brand new Robotic Positioning System [5] , that will replace the plate system and reduce target-mapping time from 20 to 2 minutes. Also, an Integral Field Unit will be installed, collecting millions of spectra with a great resolution. That new "Local Volume Mapper" is expected to map spectra with sub-parsec spatial resolution in the Milky Way, with similar techniques that are used with the instrument we mainly use in our thesis, MUSE.

[3]`https://www.sdss.org/instruments/camera/`

[4]`https://www.sdss.org/dr16/spectro/sspp/`

[5]`https://www.sdss.org/future/technology/`

2.2.2 The Multi Unit Spectroscopic Explorer - MUSE

The Multi Unit Spectroscopic Explorer (MUSE) instrument is installed at the ESO Very Large Telescope UT4, used by various surveys.

The MUSE instrument philosophy is different from the SDSS one. SDSS makes a first multi-band imaging capture, identifies objects on the obtained images, and individually collects spectroscopic data for pinpointed location, resulting in high-quality spectrum.

On the contrary, MUSE is an Integral Field Spectrograph, that can realize both imaging and spectrometry in a single pass. The instrument targets a portion of the sky, and slices the region in 24 areas. These areas are sent to 24 identical Integral Field Units, that are devices with the capability to further divide images and to capture spectra for each individual region. In MUSE, IFUs disperse each of the 24 regions into 48 “slits”, that are then analyzed using spectrographs.

At the end of that process, one obtains a 3D image of the sky, where each of the 1152 (24IFUs $\times$ 48slices) spatial pixels are associated to spectra ranging between 480nm and 930nm. Thus, at each pointing, the instrument produces what can be seen as an array of spectra — or, a 3D image — that we call a IFU product, or a datacube. These different pointings can be staked and fused together, in mosaics, to obtain spatially larger cubes. To recover a white-light image from the observation, the product can be transformed back by summing its components along the wavelength axis.

To have proper data from an observation, raw data have to be treated. The MUSE pipeline recipes software is able to automatically perform observation data correction and reduction, dealing with the bias, dark and flat fields, and also correcting the line-spread function of the instrument [49]. MUSE recipes can subtract sky emission , but one should consider performing additional sky treatments, such as the ones proposed in the ZAP package [44]. This additional processing step is often done on various spectrum products, and may be important for automatic detection and classification processes (see subsection 6.1.3).

MUSE, in its wide-field mode, has the ability to observe a $1' \times 1'$ section of the sky. As it is a ground instrument, it is affected by atmospheric effects, that can be corrected using adaptive optics.

Using all these capabilities, an IFU can be used to obtain spatially resolved spectra of extended objects. For example, the MaNGA IFU of SDSS has the capability to capture spectra from different arms of a single nearby spiral galaxy.

MUSE is originally dedicated to search for characteristic hydrogen emission lines, emitted by young redshifted galaxies. Mapping these objects permits to generate a spatial and temporal map of the Universe, and allowed new discoveries. The Hubble Ultra Deep Field Survey, leaded by Roland Bacon, observed the Hubble Deep Field [5], combining 10 datacubes obtained during 137h spanned over two years. The mosaic permitted securing, spectroscopically, redshifts from more than 1300 objects in the field, to study and constrain cosmological parameters such as reionization, but also to spatially resolve stellar kinematics of $0.2 < z < 0.8$ galaxies.

2.3 FITS files

In this thesis, we work on source detection and extraction in astronomical images. The Flexible Image Transport System (FITS) has imposed itself as a standard output format from telescopes, and has numerous interfaces across different programming languages.

Our tool will be dedicated to the analysis of these files, so we choose to add a small description and comments of these here.

For 2D images (also called broad-band images), tools such as DS9 can interpret and represent the data contained in the files, permitting visual inspection. GAIA-Skycat tool include analysis tools, and permit generating and representing catalogs automatically from FITS images.

FITS files standards are described on the NASA dedicated website [6], and is ratified by the International Astronomical Union. The authors exhaustively list all keywords and structures of standard files, we will here describe the functioning of FITS files more organically.

A FITS file is a collection of *extensions*, called Header and Data Unit(s). There must be at least one, and there is no formal limit on the number of extensions in a file. Each HDU contains a header and optionally data. The data block of a HDU is an array of values, that can have as many of dimensions one wants to. An extension containing a spectrum will be 1 dimensional, while an image can have two dimensions. An integral field unit data, such as the data produced by MUSE, will be three-dimensional. The number of axes has a virtual limit of 999 (that is, data in 999 dimensions). For observations issued from telescopes, this limit is hardly reached.

The information describing the data array is stored in a Header. Headers, written as ASCII text list, have the following structure: `KEYWORD=VALUE / COMMENT`. These lists can be accessed without loading data, and a quick look at them can be useful to know what a FITS file user is dealing with. The basic keywords in the header blocks concern axes. First, the `NAXIS` keyword describes how many axes the data array have, while `NAXISn` indicate the length of each axis, indexed with `n` varying from 1 to `NAXIS`. Other keywords concern types of data. FITS can store not only images, but also tables of data. This is set by the `XTENSION` keyword, with values `IMAGE`, `TABLE`, and `BINTABLE`. The latter stands for "Binary Table", and is similar to the `TABLE` one, except that data in it is not stored as ASCII characters but in a binary way, providing more compact HDUs and allowing to store arrays of values. Tables extensions are useful to store and handle catalogs of objects, but we will not describe their specific keywords here. Additionally, python tables are easily convertible to FITS table via the work of the Astropy project.

We'll now describe images, and more specifically, critical parameters permitting to link pixel axes to a physical coordinate system, a World Coordinate System. There are a lot of keywords reserved to that end, all standard ones described in the FITS standard paper. We'll study a few of them here.

A first keyword that will be useful to interpret data is `BUNIT`, representing the units of numbers within data array. Despite the importance of such an information, that every physicist relies on, we encountered several times FITS images without that information. Some tools rely on units to process data, and wouldn't work without.

A second set of keywords are the axis names/types. Celestial axis types are mapped through the `CTYPEi` keywords, where i is the pixel-axis to be mapped against. Classically, right ascension `RA` and declination `DEC` are used, associated with a projection algorithm as described in [15] . RA and Dec are nothing different from latitude and longitude, referring to an orientation related to a reference frame defined by Earth equator, Greenwich Meridian and earth center. `CTYPEi` comes with a reference frame, `RADESYS`, that define the axis and origin of the frame. It can be Earth system, but one can also define the International

[6]`https://fits.gsfc.nasa.gov/fits_standard.html`

Celestial Reference Frame, based on distant observation of quasars[7]. Doing so, the coordinates become independent of Earth rotation and from time. The ICRS is defined as the solar system barycenter at its origin, and axes/ecliptics are defined relatively to position of the distant objects.

Having defined `RADESYS` and `CTYPE`s, one has to define how the variation along a pixel axis affects physical axis. There are different ways to do that, because the definitions changed with standard evolution. We will not put all definitions here, but only the most recent ones.

The keyword `CRPIXi` defines the reference pixel along the axis i, and is associated with the physical value `CRVALi`. The latter value is associated with a unit `CUNITi`.

Then, the changing pixel coordinates along the various pixel axes, and how that affects physical coordinate, is defined by matrices. The most recent formalism is defined by the `PCi_j` matrix, associated to the `CDELTi` keywords defining scales. To determine the celestial coordinates with that method, two successive transformations are applied (from FITS standard paper):

$$
\begin{aligned}
q_i &= \sum_{j=1}^{\text{NAXIS}} m_{ij}\,(p_j - r_j) \\
x_i &= s_i q_i + o_i
\end{aligned}
$$

where i spans over the physical-celestial axes, j the pixel ones, and r_j is the reference pixel of the pixel axis j - `CRPIXj`. The matrix m_{ij} is the matrix `PCi_j`, and the scale `CDELTi` is represented by s_i. Finally, o_i is the origin of the coordinates, that is `CRVALi`.

Another way to map pixel to celestial axis is to use the ancient formalism of the `CDi_j` matrices. The equation transforms to:

$$
x_i = \sum_{j=1}^{\text{NAXIS}} (s_i m_{ij})\,(p_j - r_j)
$$

where the scale is integrated to the sum.

Generally, the matrices of FITS files produced by the MUSE instrument are diagonal, meaning there is no effect of rotation and distortion when moving along celestial axis. This means that a change of coordinate pixel in one direction will only affect one direction along the physical coordinates.

However, in this thesis, we are not only working on a two-axis system, but we are also interested in the physical coordinates of a third -spectral- axis. The spectral `CTYPEi` keyword can take various values, from table 25 of FITS paper, reproduced in Table 2.1.

Implementing a third axis add a column and a line to the pixel-to-physical matrix, as well as a reference pixel and value to the FITS extension header. One would also need a unit to that spectral axis When analyzing the spectrum part of a cube, it is truly important to distinguish air wavelength (`CTYPEn= AWAV`) from vacuum wavelength (`CTYPEn= WAVE`). The choice of a spectral reference system is also critical, using the keyword `SPECSYSa`, to specify if heliocentric velocity has been corrected, for example.

We must stress out the importance of having consistent keyword to define WCS. Specific non-standard keyword exist (describe the HIERARCH) and are widely implemented.

[7] `http://www.sirgas.org/fileadmin/docs/GGRF_Wksp/20_Heinkelmann_2019_ICRS-ICRF_overview.pdf`

[8] `https://fits.gsfc.nasa.gov/standard40/fits_standard40draft1.pdf`

Table 2.1: Reproduction of table 25 from FITS standard paper latest revision[8]. Presents the different codes for CTYPE keywords.

Code	Type	Symbol	Assoc. variable	Default units
FREQ	Frequency	v	v	Hz
ENER	Energy	E	v	J
WAVN	Wavenumber	κ	v	m^{-1}
VRAD	Radio velocity	V	v	$m\ s^{-1}$
WAVE	Vacuum wavelength	λ	λ	m
VOPT	Optical velocity	Z	λ	ms^{-1}
ZOPT	Redshift	Z	λ	...
AWAV	Air wavelength	λ_a	λ_a	m
VELO	Apparent radial velocity	v	v	$m\ s^{-1}$
BETA	Beta factor (v/c)	β	v	...

However, the standard WCS system is consistent when looking at physical coordinates, and a few software programs don't use them properly. For example, the Marz tool makes uses of the non-standard Boolean keyword `VACUUM` to check if the wavelength are vacuum shifted or not — while this can be done simply by checking if the spectral axis `CTYPE` is `AWAV` or `WAVE`. The same story happens while looking if heliocentric velocity have been corrected, that Marz checks by looking at the `DO_HELIO` boolean, that could be done looking at the `SPECSYSa` keyword.

Source and history of a FITS file are traced by a small set of keywords, and might also have influence on the WCS computation. The `DATE-OBS` keyword describe the date of an observation, and must not be confused with the `DATE` one, that gives information about when the FITS data unit was created. The keywords `OBSERVER`, `TELESCOP`, `INSTRUM` are transparent, and the keyword `OBJECT` should contain the name of the observed object. `EXPTIME` represent exposure time of the observation.

Fields associated to the `HISTORY` keyword traces all modifications of the FITS file, and can be used to indicate if skylines have been subtracted from a spectrum, if different images have been merged, and so on. The `COMMENT` keywords are, well, comments.

Additionally, the European Southern Observatory developed a non-standard set of keywords to further precise FITS data source : the HIERARCH convention. FITS header keywords are limited with eight characters, and HIERARCH permits to overpass that threshold; nowadays, these keywords are recognized in most of FITS files readers and parsers[9].

ESO uses these HIERARCH keywords to provide additional information on observations, and about parameters of telescopes [10].

Finally, other non-official keywords (and their associated values) are numerous. As we discussed earlier, Marz introduced some of these, despite existence of "official" keywords with the same end. Python package Astropy, and its FITS interface, does its best to read and to standardize the FITS files. Astropy also adds some non-official keywords to complete the FITS interface, such as the `UTYPE` one, that indicate the uncertainty type of data — in an uncertainty extension, that can be either standard deviation σ, variance σ^2, or weights $\frac{1}{\sigma^2}$.

[9]`https://docs.astropy.org/en/stable/io/fits/usage/headers.html`

[10]`http://www.eso.org/sci/software/esomidas/doc/user/18NOV/vola/node116.html`

Chapter 3

Astronomical images

3.1 What is an astronomical image?

In this study, we work on astronomical images, and on Integral Fields Units products — also called datacubes. An astronomical image is compound of photons collected by a telescope. These photons will produce a signal in a sensor, a camera, at the end of the observing device. There are many types of sensors, used in different spectral bands [32], and for the UV-to-Infrared bands, astronomers mainly use Charge-Coupled-Devices — CCDs. SDSS telescope and MUSE, quoted earlier, use CCDs in their devices.

A CCD is composed of a semiconductor, usually silicon. Impurities, like a few atoms of Gallium (or Boron) and Phosphorus, are added to the semiconductor crystal, in order to dope the semiconductor. That dopage increases the number of free electrons in the layer. During an observation, a photon is collected by the telescope, and will react via the photoelectric effect with the semiconductor in the CCD, creating a photo-electron and its associated hole. This photo-electron will be trapped in an electric potential well and the hole will be guided to the ground. When the observation will be completed, the CCD pixel array is read. Electrons are moved, line by line, like on a conveyor belt, to a charge amplifier, that will convert the number of electrons contained in a pixel to a voltage, an electric signal, proportional to the number of trapped electrons. Once that list of voltages (one voltage per pixel in the CCD) is collected, it can be processed written in a digital format, like in a FITS file [1].

Of course, CCDs do not work perfectly, and must be taken and processed carefully to reduce errors and increase image quality — especially when one wants to observe with a long exposure time. The quantum efficiency of a CCD measures the ratio between photo-electrons against incident photons, and denotes that all photons collected by a telescope won't be saved. A CCD also has resolution issues : the smallest spatial unit is a pixel, and not a single point. Thus, a CCD camera can limit the final resolution of an image. In the spectral domain, a CCD can't register photons that are not close to the visible light, $400 - 800$nm, even if this can be modified by adding doping atoms.

Images created by a CCD camera must also be corrected relatively to the noise created by the detector itself, in order to produce accurate output images. Different noise sources must be taken into account. When a CCD is read, a few electrons might appear (or disappear) at each pixel read, that is the readout noise. The readout noise has a normal distribution. When a pixel containing 2 electrons is read in a CCD with three-electrons read noise, there's a chance it will be interpreted as containing a negative value of charges,

[1] `http://www.if.ufrgs.br/~marcia/ccd.pdf`

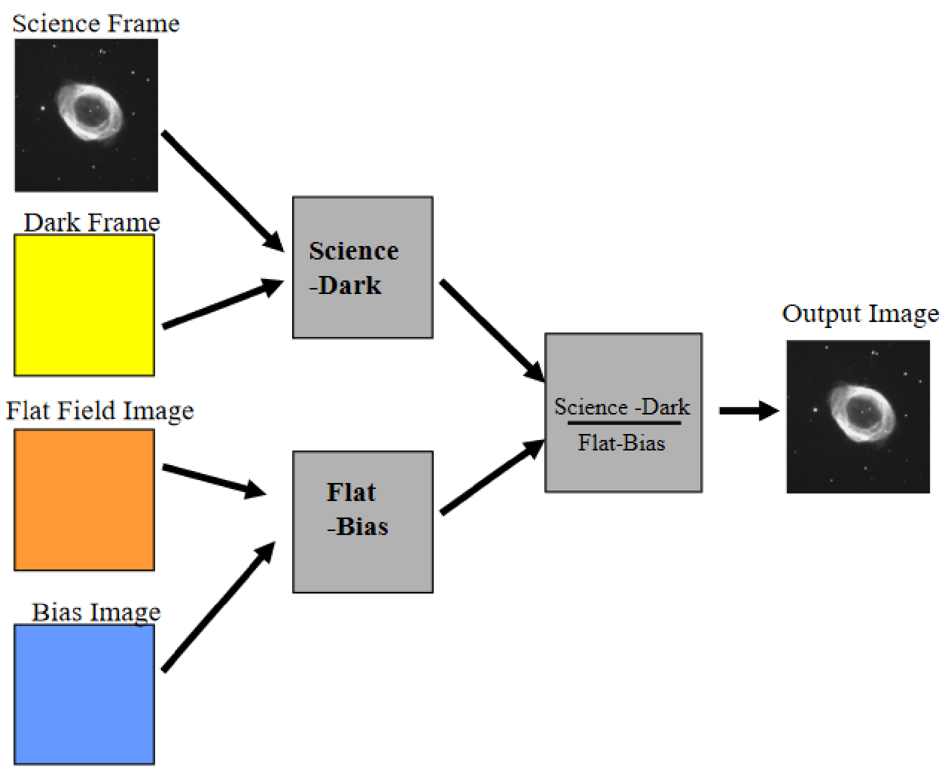

Figure 3.1: Processing science images: list of operations. Issued from `http://123.physics.ucdavis.edu/week_7_files/CCD_imaging_lecture.pdf`

.

and that would cause problems when converting it to a digital unit. To avoid that, a bias is added to each pixel, and it must be removed from the final frame.

To estimate bias and readout noise, one can take several images, with 0 exposure time, and the sensor plunged in the dark. The mean of various images will be the bias field, while the standard deviation between the frames would give an estimate of the read-out noise.

A second issue is photo-electrons induced by the temperature of the CCD chip itself, the dark current. It can be reduced by cooling the device, for example MUSE is cooled by liquid nitrogen at $-130°$C. To measure it, one can take a few images with exposure times identical to the future observations, but with the shutter closed, hence observing the “dark”.

The various pixels of a CCD chip may have different efficiencies. To qualify that efficiency map, flat fields are measured : images taken of homogeneously illuminated areas.

Once all these images have been taken, on can finally collect science, or observation, frames, and correct them by applying operations in Figure 3.1.

Finally, the uncertainty, or the noise, associated to a fully processed image is:

$$\sigma_{total} = \sqrt{(\text{DarkNoise})^2 + (\text{PhotonNoise})^2 + (\text{ReadNoise})^2}$$

The dark and photon noises are following Poisson statistics, meaning that the uncertainty associated to a measurement is :

$$\sigma_N = \sqrt{N}$$

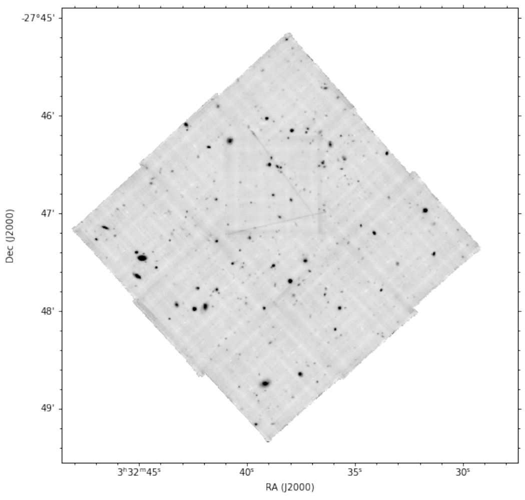

Figure 3.2: Hubble Ultra Deep Field negative of the white light image, obtained from a MUSE datacube using `AutoMUSE`.

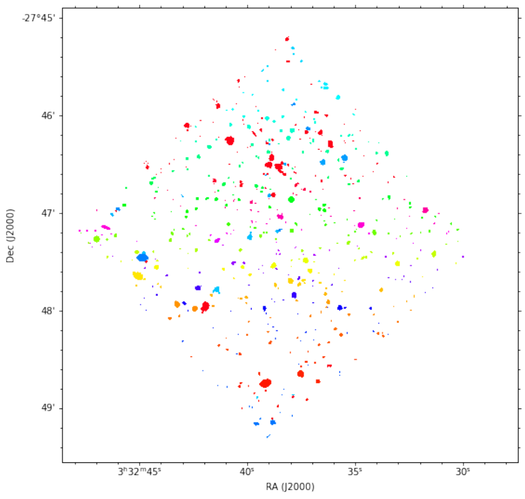

Figure 3.3: Hubble Ultra Deep Field segmentation map, obtained from a MUSE datacube using `AutoMUSE`. Each colored region represent a different source, as detected by the tool.

, where N is the number of events — the number of detected photons, in our case.

Images taken from professional telescopes usually flow through a dedicated pipeline, realizing these steps automatically, resulting in FITS files containing two extensions, data and its associated uncertainty.

The MUSE instrument and its pipeline produce 3D FITS files. The third dimension comes by splitting the captured flux in different light rays, and by projecting these onto CCD spectrographs. We then have a count of photons per wavelength unit.

3.2 How to detect objects on an image

In this thesis, we are working in producing "segmentation maps". Segmentation maps are bit-masks of an image, in which the background and identified features of an observation are disambiguated, and where each different feature is labeled with a distinct integer, like on figure 3.3.

A segmentation map can act like a mask, and using it, one can select one-by-one objects in its original image. Contrary to a catalog, that only store centroid coordinates, they can store the shape and topology of an object in the sky. Nevertheless, two celestial sources can be blended, or even fully covering each other while projected in the sky, and that cannot be represented correctly on a segmentation map, while a catalog can store two objects in the same place. According to SDSS statistics[2], about 15 to 20% of objects are partially blended in the sky, thus a proper deblend is a rather important step.

To produce a segmentation map, one must start by finding objects on an image. As we work not only on images, but on datacube, we will show some ways to extent image detection to datacube detection.

There are various ways of detecting features on an image. In the era of machine learning algorithm, we can train algorithms into recognizing faces, or common-life objects,

[2]`https://www.sdss.org/dr12/algorithms/deblend/`

in images. However, to train such an algorithm for astronomical images one would need to have a repository of segmentation map and white-light images, and to work on different sets of algorithms to select the most appropriate one for such a task. This would require an extensive computer time for the algorithm to learn what is an astronomical object, and additional analysis to distinguish false positives in the detected objects sets. That technique would also require to actually have a repository containing segmentation maps and images.

Another way to perform detection is the approach used by Origin [34] to detect emission-line objects. This tool is based on various algebra, to estimate the background and remove "nuisance", and pattern recognition techniques, to recognize the spatial-spectral shape of an emission line.

Nevertheless, simpler, standard astronomy imaging techniques exist in order to obtain a segmentation maps of an image. One of the most used - and fast - tool is SExtractor. Other detecting techniques for astronomical image are summarized and described by Masias et al. in [35].

SExtractor

SExtractor [9] is a widely used tool in the astronomical field. Taking as input a FITS image and a parameter file, it is able to find objects on an image, and to produce a catalog of measurements (including photometric ones) and of coordinates. It is also able to produce, as a by-product, segmentation maps.

The library retrieves various information in headers of the FITS file (such as gain, units in the image, Right Ascension and Declination) to produce its catalogs. However, detection parameters are not affected by these, and are impacted only by the configuration files. If we are interested in the core of SExtractor, image detection, here is a way of explaining what it does. A majority of parameters descriptions can be found in SExtractor for Dummies [25] .

SExtractor works with an ASCII parameter file, where each line is written as:

PARAMETER   Value(s)

(If there are different values for a parameter, they must be separated by commas).

In a first place, SExtractor imports the FITS image, and computes its **background**. To do so, it draws meshes of sizes specified by the user using parameter `BACK_SIZE` (default size is 65px × 65px), such as on figure 3.4.

Within each of these meshes, the tool computes a histogram of values. Recursively, each pixel value is discarded from that histogram when its value is not within the mean $\pm 3\sigma$ — and the statistics are recomputed until no values are discarded. If the standard deviation σ value dropped by less than 20% at each iteration, the software considers that the field is not crowded, and use the mean value of the histogram as local background value once it stopped its iterations. In the crowded case, the resulting pixels values from the histogram are used to compute the background in a local mesh, using the equation

$$\text{Backgound}_{\text{value}} = 2.5 \times \text{median} - 1.5 \times \text{mean}$$

The background values are then median-smoothed over the mesh-image (where each pixel represents the value of a mesh) using the filter size stored in `BACK_FILTERSIZE` parameter.

The median-filtered background mesh-map is then resampled to the image size using a bi-linear interpolation, according to the original paper [9], or a bicubic spline one according to the online documentation [36]. In parallel, a background "Root Mean Square" map is computed — that is, per mesh, the standard deviation of the background noise.

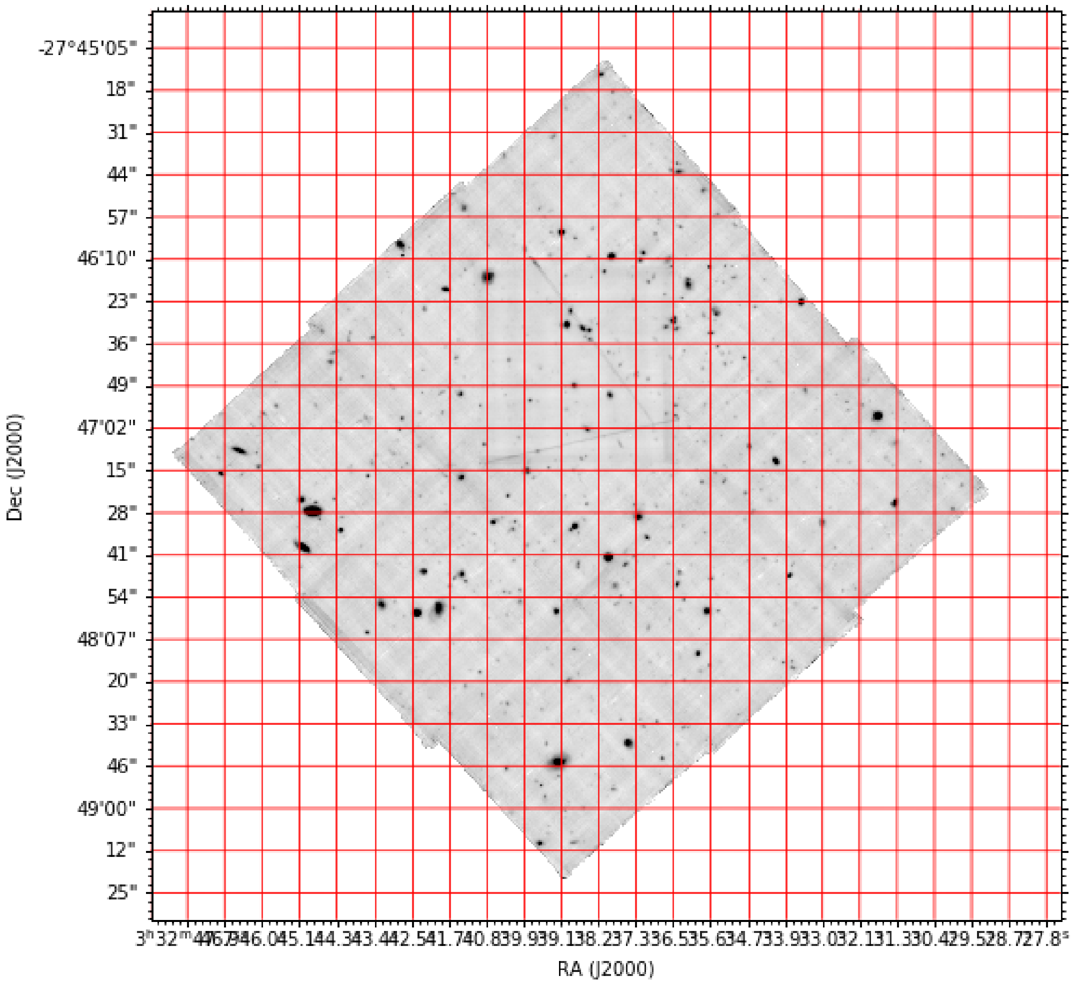

Figure 3.4: Hubble Ultra Deep Field white light image, with a grid of mesh-size 65 pixels. In each of these meshes, a different background value is computed.

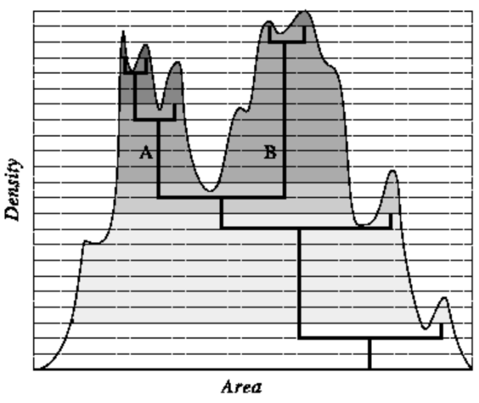

Figure 3.5: Figure 7.4 from [25]. SExtractor builds a tree of pixels intensity in connected regions, and uses its branches to deblend objects.

Then, according to Source Extractor for Dummies [25], a threshold-filter is applied to the data image. All pixels that are above Background $+ t \times \sigma$, where t is the threshold set in `DETECT_THRESH`, and σ being from the RMS-map.

These detected pixels are analyzed, and the ones that are not in a region of more or equal to `DETECT_MINAREA` connected pixels are discarded. The pixel connection in SExtractor is a 8-connection, as shown in figure 7.3 of [25].

It will also deblend the objects that are close to each other, by building a “tree” of the pixels’ intensity, and separating the branches as different and individual objects. This behavior can be set and parameterized via the configuration files.

Once objects are deblended, SExtractor is able to produce a segmentation map of the image, that will be written directly to the disk as a FITS file, if the parameter `CHECKIMAGE_TYPE` is set to `SEGMENTATION`. The software also generate a catalog, with the approximated center-of-light pixel coordinate of every detected celestial body. After performing object detection, SExtractor can extract photometric and morphological information from each detection, and write that to the issued catalog, that can be analyzed by any tool able to read ASCII files, or FITS files.

One of the reason that make people use SExtractor is quoted in Sextractor for dummies [25] : the tool goes *fast*. With its default parameters, it is also able to generate accurate results, if your image is bright enough. The tool is robust, and made its proofs, with more than 7400 citations of the original article [9]. Additionally, it is resilient against clumsy parameters and FITS files with wrong header information, meaning it will produce (probably incorrect in the photometric part) results without returning an error code, and without catastrophic failure. The tool is also embedded in astronomical software collections, such as Scisoft or Starlink, making its use versatile.

It is used for detecting stars and galaxies in not-too-crowded fields, but also fainter objects, such as asteroids [42].

It is also used to detect faint, emission line emitting galaxies, with algorithms such as MUSELET [37].

3.2.1 MUSELET

MUSELET (MUSE Line Emission Tracker) [37] is a python tool in the MPDAF package, dedicated to analyze MUSE products.

The goal of MUSELET is to extend the capabilities of SExtractor to analyze MUSE datacubes, as the mentioned tool can only work on 2D images. The tool handles detection against the third, spectral, axis. To analyze a MUSE product, the script loads it, and cut it along its spectral axis every 5 wavelength pixels. In the physical sense, every wavelength pixel of a cube issued from MUSE is 1.25 angstroms wide, thus the obtained bands are 6.25 angstroms thick. These numerous narrow-band cubes — a MUSE datacube has $\approx$ 3600px in the spectral direction — are spectrally averaged to narrow-band images, and written to the disk. These narrow-band images are completed with RGB images, that are images produced from $\frac{1}{3}$ of the cube in spectral direction. All are analyzed with default SExtractor parameters (if the user have not specified any parameter file), and a catalog for each is issued. All objects centroids, from all these catalogs, are placed in a k-dimensional tree [29]. Using this structure, the nearest neighbors of each centroid, at a distance of 5 (spatial) pixels maximum, is searched, and the centroid of every "ball" of points is computed. These points represent the spatial coordinates of objects, as detected by MUSELET.

As a result, the user obtains that list of coordinates, associated to positions of potential emissions lines. The algorithm tries to associate a redshift based on the positions of these lines.

3.3 Limits and critics, development of a different program

The goal of this thesis is to provide a script with the ability to generate a reliable segmentation map of a MUSE datacube, through an easy to install, to use and to modify Python script. The parameters must be understandable, and to accord a certain flexibility to the user. It must be able to produce a "confidence" level for each detected object in the segmentation map. To produce such a tool, one could think that we could start with the MUSELET algorithm as a basis, and extent its capabilities into producing segmentation map with SExtractor, and then combine them. Originally, we tried such an approach, but had to fix many issues one by one, gently departing from the original algorithm, and leading to a completely different one. The first problem is related to SExtractor behavior and internal functioning. SExtractor has no python interface, and all parameters must be written to a configuration file or in a command line, and must be launched from command line, making it os dependent. There are attempts to port SExtractor to a python package, as a wrapper or as a module, such as sewpy [45] or SEP [7]. Even using these modules, the SExtractor workflow is not *pythonic*, producing strangely organized codes for python users.

A second problematic with SExtractor is that it can't work with memory-stored arrays, it only works on FITS files written to the disk. If one want to analyze FITS data (generated by astropy, for example) using SExtractor, s.he has to write it into a file, analyze it with SExtractor — that will then load it back to memory, do some computations, write results to the disk — and load results back to python memory. This is fine for small-scales analysis, with a few dozens of files, but will become inefficient while using up to a thousand files.

Moreover, SExtractor main functionality is to produce catalogs, not segmentation maps. These last ones are a by-product of the tool, so it seems justified to use another tool, more dedicated to that production.

Finally, a last problem comes from MUSELET. The package is great to handle the Muse products, but is inefficient in terms of memory use in the scenario of our thesis.

Indeed, MUSELET, while loading a datacube — that can weight several gigabytes — will store it twice in the computer RAM. Moreover, it uses SExtractor, rendering a program with a strange workflow, as stated above. Finally, the tool is not aimed to segment a datacube, but to find object centroids in all wavelength-layers of an input file. The detected centroids, if close to each other (see the above section) are considered to be belonging to the same celestial body — which is not the case for close and not fully deblended objects. The software, even if inspiring in its methods for our thesis' goals, is not suited to produce segmentation maps.

We thus choose to implement our own tool.

Chapter 4

AutoMUSE

4.1 Basis of the script

We want to produce an easy-to-use and easy-to-read tool, that would run reasonably quickly. We choose to use Python 3, for its versatility and for its numerous science libraries, that often use C and C++ under-the-hood - such as NumPy [23], and Pandas [41]. It also has useful packages oriented towards astrophysics — astropy [2, 3]— and data analysis — that are too many to be cited, but the first one to come in minds is scipy [48]. Our script, called Autonomous MUSE Source Extractor (AutoMUSE), is based on the following python dependencies :

Python 3.7 (or above): core of the program, programming language.

astropy 4 (or above): to handle FITS files in python, and astronomical data;

scipy : for signal and image processing;

pandas : For tables management;

numpy : for numpy;

multiprocessing : to handle parallelization of Python functions;

subprocess : to run external programs;

tqdm : to display progress bars;

photutils : for image processing [12];

pint : to handle units until astropy.units is improved [20].

To use the automatic classifier (attempt) capacity of the script, one also needs to install **Alex Griffiths' Marz** version [24][1].

AutoMUSE needs a few arguments to function, and they will be described through the section:

fitsfile : your datacube to analyze;

–output : the output directory, where all files and results will be produced;

[1]`https://a-griffiths.github.io/Marz/#/overview`

–n_cpu : the number of CPUs/processes you want to use;

–delta : the number of pixels along wavelength-axis a narrow-band image will be composed of — see subsection 4.3.1;

–step : Step selection, if the program has already been started and that you don't want to re-run all steps. If `int`, like "`0`" (without quotes), the step at which the script should start; if `list`, like "`3,7`" (without quotes), the steps the script should execute.

`AutoMUSE` can be launched in a Command Line Interface through `python3 automuse.py` *`args`*.

4.2 Reading FITS files from Muse

To be able to process data, reading it is a necessity. The FITS standard in astronomy provides a set of rules (see section 2.3), incorporated in different FITS readers/writers that exists in the scientific world, and evolving across versions of telescopes or software. Ironically, the FITS standard is not a standard, as every team and facility add their own layers.

To simplify the problem, we propose to use reformat the input data to make it compatible to the astropy.nddata.CCDData object.

This class is designed to process CCD and image products in astropy, and handle units, uncertainty, masks, flags, and World Coordinate Systems associated to data contained in FITS files. It is especially useful to select a chunk of data in a cube, as it will automatically generate new WCS on-the-flight for the new chunk, as well as selecting the correct part of the uncertainty array. An additional interesting feature of the CCDData class is its ability to propagate uncertainties along calculations, but we won't use that feature in our program. Using CCDData allows astropy to automatically correct and standardize *most-of* WCS header keywords. In this section, we describe our method to make a formatted copy of a datacube, in order to obtain, as a final product, a FITS file easily readable by the class mentioned above.

In a first place, we find the extensions containing data — a 2D image plus a spectral axis — and uncertainty associated to it from an input FITS file.

According to the MUSE pipeline [49], a datacube should present two extensions : DATA, and STAT, containing variance, so this step should be optional. The primary (DATA) header would contain information about WCS: right ascension, declination, units ... However, analyzing various cubes leads us to notice the *great* variety of extensions names and format presenting the two considered data arrays. We thus chose to do the following :

- If the first extension of the datacube contains an array, it is considered as the data array.
- If the first extension does not contain data, the program search non-empty extensions with names in "DATA", "FLUX", "INTENSITY", and consider the first detected as data.
- Data variance extension, "STAT", is searched through non-empty extensions named "STAT", "VAR", "VARIANCE", "WEIGHT", "STD", "DEV", or "UNCERT".

If the STAT-type extension is not found, the rest of the program should work, but results would be less-accurate.

Then, the formatting function parses units of both data and stat, using the `pint` library instead of `astropy.units`. The `pint` module has proven itself more reliable to figure out the correct units under the `BUNIT` keyword of the FITS file headers, that sometimes causes errors while parsed with astropy. We will update the script function when the units module of astropy will prove itself reliable, in our will to work within a single coherent framework to simplify computations and reading.

For the data extension, if the units do not parse, or if the BUNIT keyword is absent of both primary and data headers, the data is assumed to be expressed in $10^{-20}\mathrm{erg\,s^{-1}\,cm^{-2}\,\mathring{A}^{-1}}$ - as in MUSE pipeline.

Then, there is an attempt to create the CCDdata instance, without further checking to the STAT extension. Astropy will try to automatically determine what is the uncertainty type based on headers. However, if this does not work, units from the uncertainty extension are retrieved, and based on them, the type of uncertainty array — is it representing a standard deviation, a variance or weights? If that determination fails, variance will be assumed. The uncertainty array will be converted to variance, and stored in the CCDData cube.

After getting through data loading, the formatting function will load, analyze and correct the World coordinate system to standard keywords and axis types. As defined by the FITS standard guidelines [40], WCS headers have fundamental components describing axis, in particular the spectral axis. The script determines if the measured wavelength of the spectrum is taken as the vacuum or air one. In order to do so, it checks if the keyword `VACUUM` is set. If not, it will verify if the `CTYPE` associated to the spectral axis is `WAVE`, `LINEAR` — for vacuum wavelength — or `AWAV` (air wavelength), and set the `VACUUM` keyword accordingly. Handling logarithmic scale is not implemented.

That formatting step will then produce a FITS file containing data and variance extension, fully compatible with astropy.nddata.CCDdata interface. Along with that file a white light image of the cube is computed and produced, with the same compatibility to the above interface.

4.3 Images, narrow-bands detection

4.3.1 Generating narrow-band images

In MUSELET [37], Johan Richard chose to cut MUSE datacubes along the spectral axis in order to detect the faint single emission line celestial bodies — such as Lyman-Alpha Emitters. According to Wisotzki and al. [50], nearly all the sky is covered by that emission, due to the transition between the orbital level n=2 to n=1 of an electron belonging to an hydrogen atom. Warm hydrogen gas, surrounding galaxies, will emit photons with wavelength 121.567nm. That wavelength will then be redshifted during its travel to Earth, and enter the MUSE wavelength coverage between redshifts $3 \lesssim z \lesssim 6.5$. Detecting such an emission line can improve faint object findings, as well as bring information to secure the redshift of a celestial structure. Figure 4.1 shows an example of a Lyman-α emitter, with a sole Ly-α line.

A single emission line structure is much more contrasted — thus easily detectable — in a narrow-band image. In our program, we decide to roll a 5-wavelength pixel (the 5 pixels length is modifiable via the argument `-delta`) window along the wavelength axis.

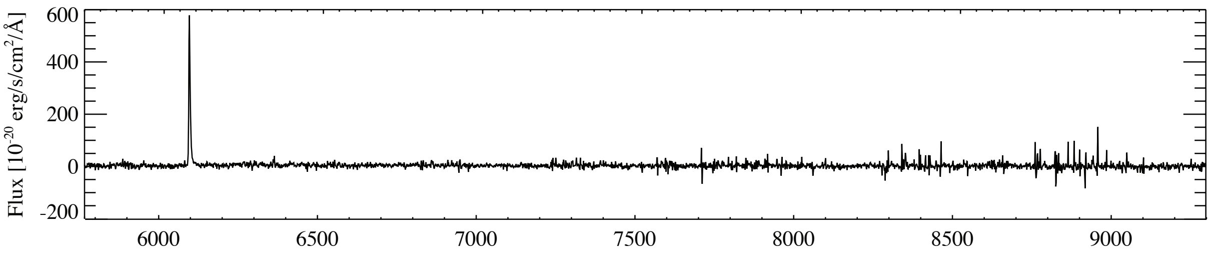

Figure 4.1: Spectrum of a Lyman-α emitter, showing almost no continuum, while having a strong emission in its Ly-α band, from Figure 5 in [4].

The 5-wavelength-pixels layers in the window are weighted with a Gaussian with $\sigma = 1.26$ centered on the central spectral layer, and summed along the wavelength axis, giving narrow-band images, as illustrated on Figure 4.2. The technique used in MUSELET is similar.

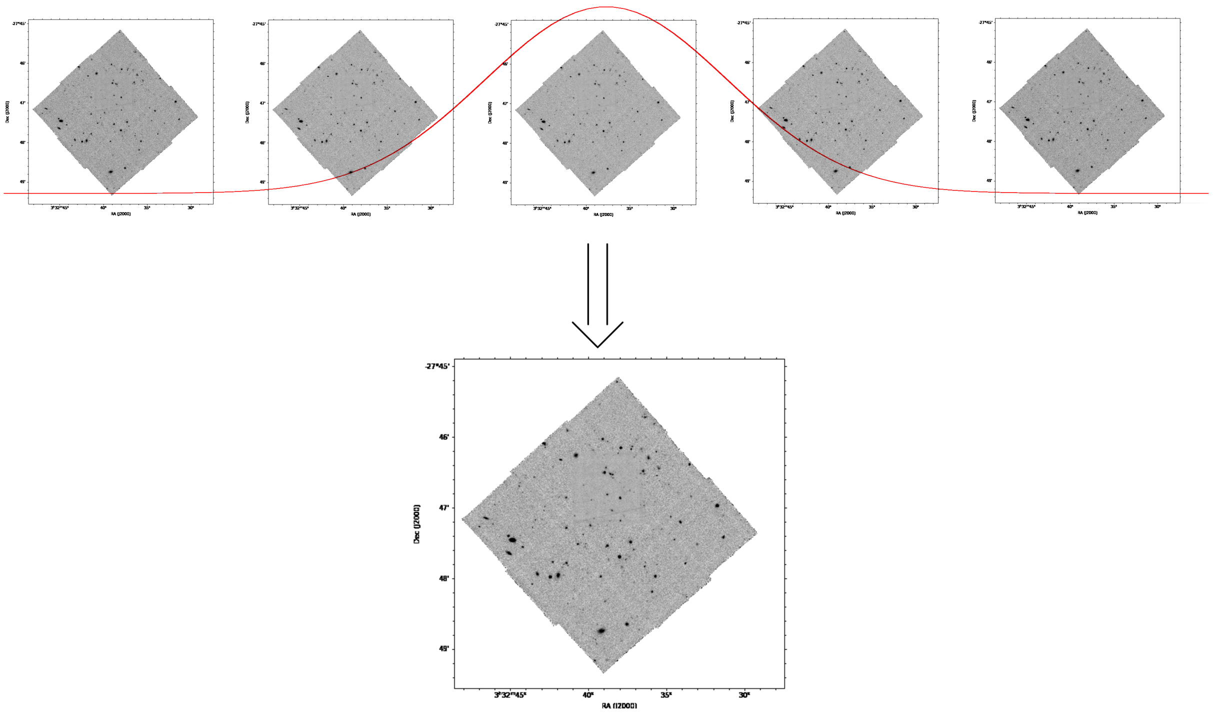

Figure 4.2: Narrow-band process illustration, taking `-delta` wavelength layers, and weight-summing them to a narrow-band.

For a complete MUSE product, this generates ≈ 3600 narrow-band images. To avoid having to compute these images several times when analyzing the same datacube with different parameters, the images are stored as FITS files. This step is parallelized using the `multiprocessing` python package, to reduce compute time.

4.4 Finding the bright pixels

In computational astronomy, a source is a subset of bright pixels in an image. To detect them, we use the same workflow as in section 3.2, through the python package `photutils` [12], to have more flexibility than SExtractor in terms of python interaction.

4.4.1 Background generation

Using the cited package, we generate a background image for each of the bands — narrow-bands and white-light. `photutils` comes with an handy tool to roughly mask the first sources in an image, `make_source_mask`. This function is parameterized in AutoMUSE to find pixels, connected to at least 5 others falling in the same category, laying 3 σ above the image pixel values mean. Mean and standard deviation are computed internally using sigma-clipped statistics, similarly to what is explained in Figure 3.2. The detected regions are aggressively extended of 11 pixels, generating a source-mask.

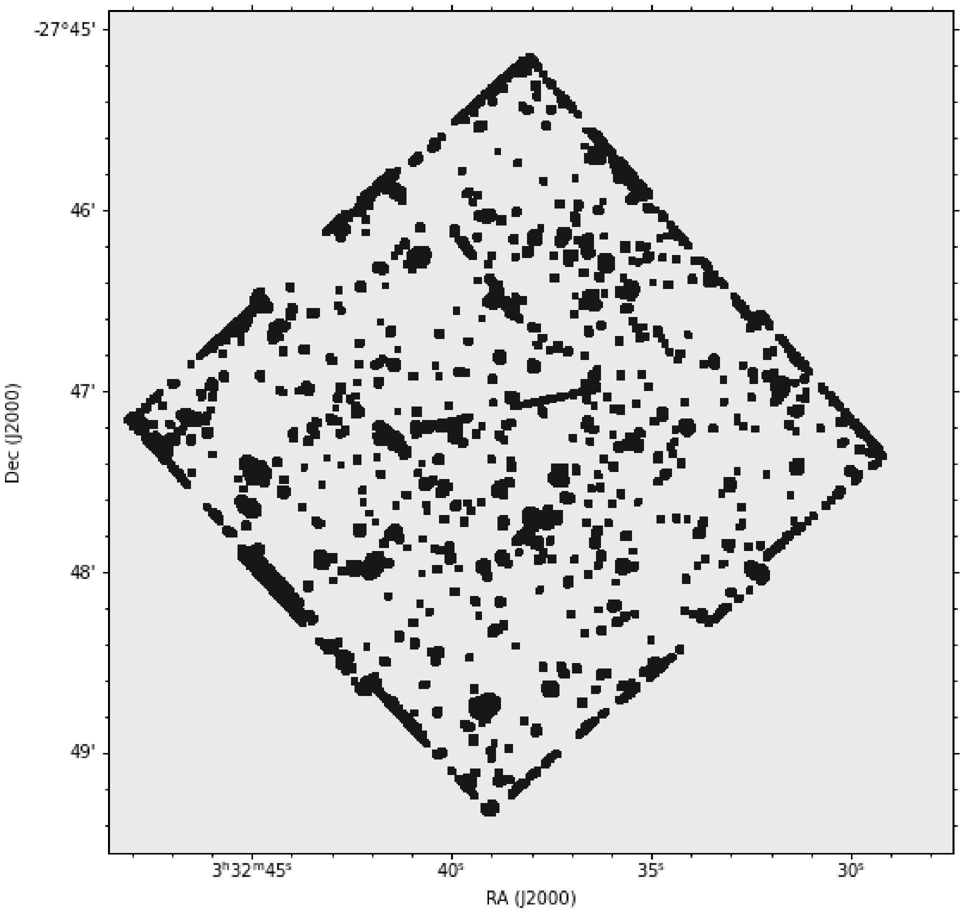

Figure 4.3: First pixel sources mask, generated by `photutils` of Figure 1.1. Each source, detected by photutils during its preliminary run, is expanded and masked.

Then, that source mask is used to generate a background image. The mask is used to hide the bright sources, and meshes of $64\times64\text{px}^2$ are drawn. In each of these meshes, pixels laying outside of $\text{median}_{\text{mesh}} \pm 3\sigma_{mesh}$ are clipped iteratively ten times. The background value of each mesh is defined as (similarly to SExtractor):

$$\begin{cases} (2.5 * \text{median}) - (1.5 * \text{mean}) & \text{if } \frac{\text{mean}-\text{median}}{\sigma} > 0.3 \\ \text{median} & \text{otherwise} \end{cases}$$

One should notice that there are different types of background-estimators, for example one can take the mean or the median in a cell as background value. More robust estimators, such as the SExtractor one, are less sensible to the presence of existing sources in cells. SExtractor uses a Mode estimator: a linear combination of mean and median in a cell is

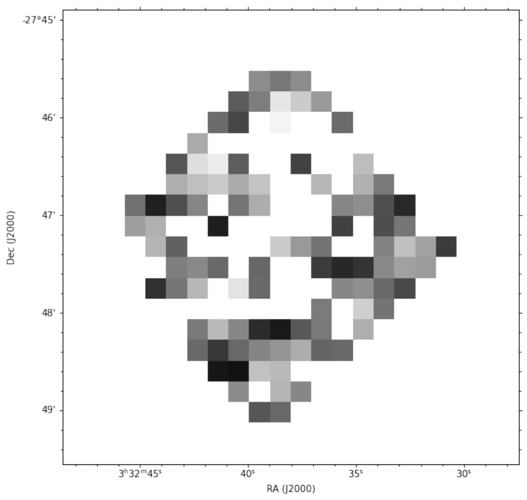

Figure 4.4: Mesh-background, generated by `photutils`, of Figure 3.2.

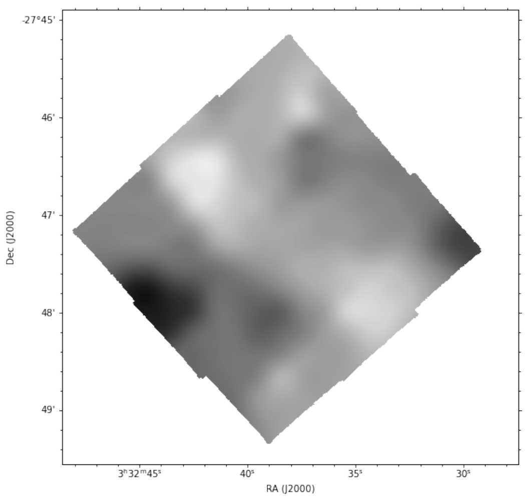

Figure 4.5: Background, interpolated, generated by `photutils`, of Figure 3.2.

used as background. Other programs, such as DAOPHOT [46] uses other linear combinations, like the MMM algorithm, a mode estimator of the form $3 \times \text{mean} - 2 \times \text{median}$. The MMM algorithm is, according to [9], less accurate when using sigma-clipped distributions of pixels. The `photutils` package, that we use to do our computations, includes these various estimators.

The mesh values (Figure 4.4) are then upscaled to the original-size using spline interpolation (Figure 4.5). Along these background images, local σ-maps (also called RMS images) are generated, first for the meshes, then interpolated to the resampled image.

4.4.2 Thresholding and obtaining the pixel sources locations

Once the background is obtained, a threshold image is generated, computed with the formula:

$$T = B + t \times \sqrt{V + B_{rms}^2}$$

where t is `-threshold` , B is the background-image Figure 4.5, V is the variance in the variance-extension of the image, and B_{rms} is the background root-mean-square. The threshold-image is convolved with a Gaussian kernel in a 3px × 3px windows, to smooth it.

Then, `photutils` compares this threshold-image do the input data. Every pixel, in the analyzed image, above its corresponding value in the threshold image is flagged as `True`. The pixels that are not in connected regions of at least 5 flagged pixels are discarded. The output is Boolean image, that will serve as basis for a segmentation image.

This process of generating background and thresholding images is repeated for each narrow-band and the white light image, creating thousands of mask images that remains to be analyzed to obtain the final segmentation images.

4.5 Fusing and deblending objects

The result of the previous step is a set of Boolean pixel maps, for each of the narrow-band and white-light images, marking the regions where pixels have been flagged as belonging

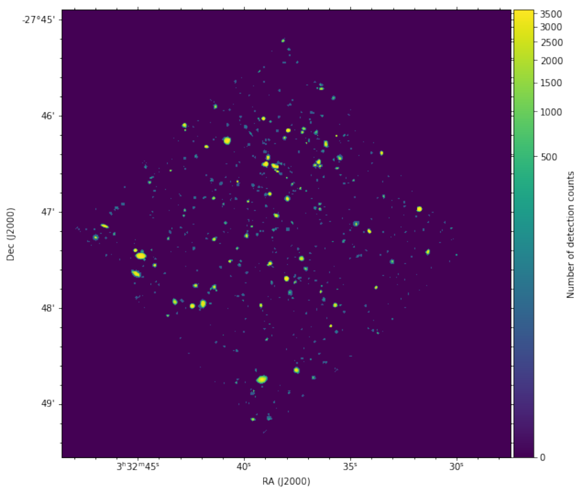

Figure 4.6: Detection-count map of Figure 3.2. Each pixel has the value of how many times it has been detected on broad and narrow-band images.

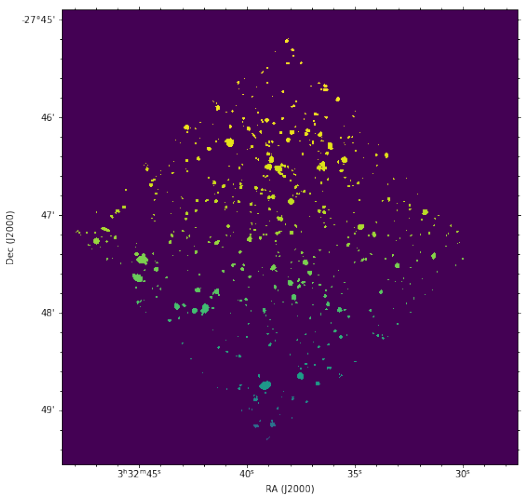

Figure 4.7: Labeled regions of Figure 4.6. Each source is assigned to a different integer.

to a source. For a standard MUSE cube, that represent ~ 3600 images. We now have to combine these maps, and separate the blended objects.

To merge the different pixel maps, we explored various options. In a first place, we tried to deblend the Boolean pixel maps individually, based on intensities and shapes. Then, the idea was to merge these maps, thanks to a coefficient quantifying the number of pixels in common. This method had several issues : it worked on relatively small sets of stacked images (10-20 images), but dramatically failed while using the entire array of narrow-deblended maps. This was due to presence of large objects on some images, that by their sizes included the smaller-deblended objects in other images, thus removing them. Moreover, this method required to inspect, individually, each of the detected region, in each of the images. To give an order of magnitude, this represents, for the Hubble Ultra Deep Field (Figure 3.2), nearly 500 objects per image with a σ-detection threshold of 3, and the method would have required to go several times through around 2 millions objects.

Then, we chose to sum the detected pixels maps, to an image that would summarize the detection count across all segmentation maps, as shown on Figure 4.6. Every region where the number of count is not null is labeled differently (Figure 4.7), and is analyzed independently.

4.5.1 Elliptic filtering

To avoid over-segmentation - deblending sources that should not be deblended — each labeled region goes through an “elliptic filter”. The script estimates moments of objects, and derives the semi-major and semi-minor axis a and b that would fit an ellipse the best to an observed label, using `scikit-image` measure function `regionprops`. Then, we compute the area of such an ellipse ($ab\pi$) and compare it to the actual area of the region of connected pixels. If

$$1.015 \geq \frac{\text{area}_{\text{pixels}}}{ab\pi} \geq 0.985$$

the object is assumed to be elliptic and not to be further deblended. This is a strong assumption, thus the results of these steps must be taken carefully. Blended objects that have an elliptic projection on the sky will not be split, and single objects that are not looking round will be spitted.

The 1.5% threshold has been chosen by trial and error, to find the correct balance between large and small objects in segmentation maps.

Thus, discrete objects laying next to bright sources will probably be absorbed, and in case of the search for such sources, one would need to inspect the resulting objects manually.

4.5.2 Number-of-detection deblend

After having filtered elliptic objects, we start the deblending process, using a watershed algorithm provided by `photutils`. A watershed algorithm can be visualized as the following :

- Local minima in an image are searched. A different color "water source" is associated to each of them.
- Watershed starts by pumping colored water in each of the sources, raising the level of water.
- When differently-colored water encounter, the watershed stop pumping, and surfaces of colored lakes are two (or more) different regions.
- When two regions have made contact, water can still be pumped in other directions, until it meet different colors.

In a first step, we apply the watershed on the previous number-of-detection image (Figure 4.6), convolved with 3×3 Gaussian kernel. To start with, pond borders are set at the edge of each labeled regions. The algorithm reverses the image — "summits" are now the deepest points of the image — , searches for the local minima (previously, summits) and starts to flood. The regions that contained different summits in terms of number-of-detection, are deblended. We set a limit to the deblended region sizes, that must be larger than 5 pixels area.

At the end of that step, we apply again the elliptic filtering, described in subsection 4.5.1.

4.5.3 Topographic-distance filtering

For each individual region derived from previous step, we compute the euclidean distance of its pixels from the borders of the object. Again, we reverse the obtained image, and apply the watershed algorithm. That distance-process is shown in Figure 4.8.

That process creates new deblended objects, that are again analyzed to check their ellipticity, to remove some of them from further analysis.

4.5.4 White-light deblend

To complete the deblending process, we now use the white-light image of the cube. This step, sometimes done using external images of the same sky square, is here done using

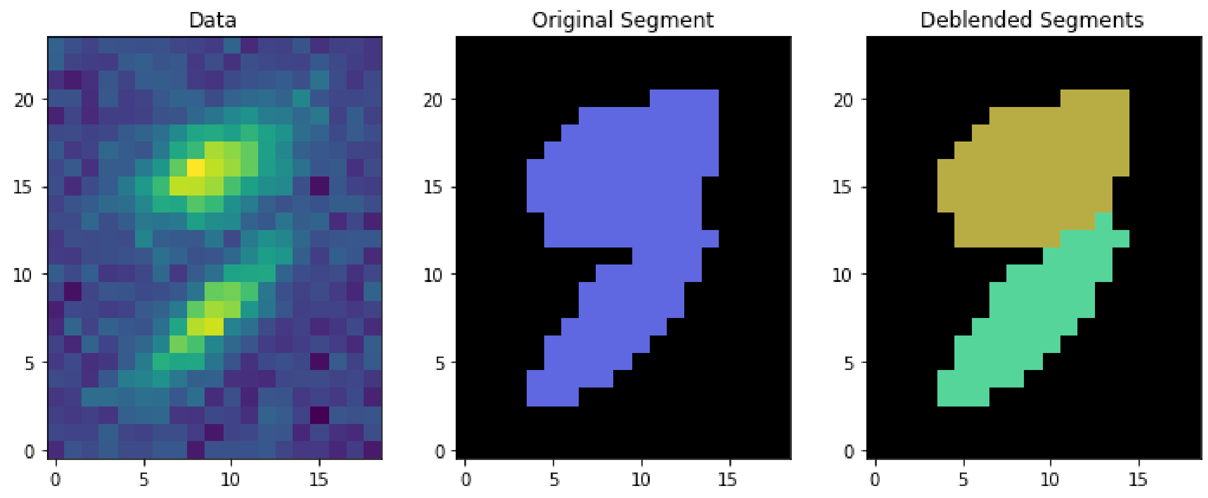

Figure 4.8: Example from photutils, deblending using topological distance.

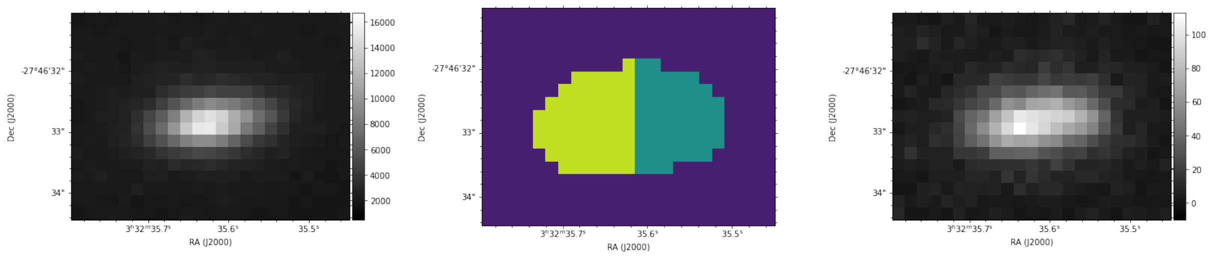

Figure 4.9: Late type emission galaxy at a spectroscopic redshift of $z = 0.52463$ found on Figure 3.2. Left panel: White light image. Middle panel : Segmentation of the object, before spectral fuse. Right Panel : Narrow-band image, centered around the brightest emission line of the galaxy.

the cube data flux summed along the spectral axis. We load the target cube, and weight-average it to its white-light image — weights being the inverse of the variance $\frac{1}{\sigma^2}$. That white-light is passed to `photutils` deblending algorithm, with the same parameters as before. We apply elliptic filter on the resulting segmentation map again.

4.5.5 Hull cleaning

All the objects that are not flagged as elliptic, or not present in the $10 - \sigma$ white-light segmentation map, are "hull-cleaned". We start by computing their hulls, the smallest convex polygon containing all pixels of each label. Then, we check the solidity - ratio between area of the object and the hull's one - of the region. If the solidity is less than 80%, the objects are discarded.

All the non-elliptic objects that are close to the borders (10 pixels) are discarded.

4.5.6 Spectral fuse - Pearson-coefficient filter

After the entire deblending process, used to separate sources, we implemented to the algorithm a part to clean wrong divisions. This part is quite important to correct the over-segmentation phenomenon. An example of over-segmentation is available on Figure 4.9.

To correct the segmentation map, we use the full extent of the wavelength axis. For each label, AutoMUSE finds if there are neighboring (touching) ones. If two labels are connected, the script computes their spectra, by summing over spatial axis the pixels contained in their segmentation map. To remove border effects, produced spectra are cosine tapered. Additionally, they are sigma-clipped — with $\sigma = 8$ — and smoothed with a Blackman window [10]

$$w(n) = 0.42 - 0.5\cos(2\pi n/M) + 0.08\cos(4\pi n/M)$$

5-pixels wide.

Then, the Pearson Correlation coefficient ρ between the neighboring spectra X and Y are computed:

$$\operatorname{cov}(X,Y) = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{1}{2}\left(x_i - x_j\right)\left(y_i - y_j\right) = \frac{1}{n^2}\sum_{i}\sum_{j>i}\left(x_i - x_j\right)\left(y_i - y_j\right)$$

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y}$$

If the regions occupied by both neighboring labels are larger than 25 pixels, the two regions are considered to be the same object if the correlation coefficient is higher than 98%. Else, for smaller objects, that threshold is lowered to 80%.

The spectral comparison produce permits correcting over-segmentation such as on Figure 4.9.

4.5.7 Eccentricity cleaner

Finally, all labels/objects are cleaned according to their extent and eccentricity. If their extent, the ratio between pixels in the objects and the bounding box area, that is the smallest rectangle area containing the object is 1, the source is discarded. Moreover, if the object eccentricity of the ellipse that has the same second-moments as the region is more than 0.98, the object is discarded, as interpreted as a misdetection.

4.5.8 Segmentation map file

The segmentation map is written to the disk as a FITS file containing two extensions. The Primary one contains the segmentation map array itself ; where each integer value represent an individual object. The second extension is called a "trust" array: each pixel value is

$$P(X,Y) = \frac{\sum_{i}^{\text{Segmentation-maps}} Seg_i(X,Y)}{\text{Number of segmentation maps}}$$

Where Seg_i are Boolean segmentation maps of narrow-bands and broad-bands : on these images, pixels where a source is spotted are equal to 1, while the others are 0. That extension must be taken just as an indicator : some emission-line galaxies are detected on 2 or 3 segmentation maps, over more than 3000 maps.

4.6 Extraction of objects and of spectra

Once the objects are found by the previous steps of the scripts, they are conveniently extracted to permit further analysis easily. For each label in the segmentation map, we draw their minimal bounding box, and expand it by 4 pixels on each side. Then, that bounding box is placed on the CCDData - formatted cube of section 4.2, and pixels within the box are extracted and put in a FITS file. Again, that FITS file respects the CCDData conventions: a data, an uncertainty and a mask extensions are present. The uncertainty extension is variance, while the mask is a Boolean array set to True where the pixels are background : it is the opposite of the segmentation map of the studied label. The source center Right Ascension and Declination are computed from WCS information, and written in the data extension header, under the cards `RA` and `DEC`. The `NAME` card represents the ID of the object (the integer of the label in the segmentation map).

Once the sources are extracted, they are reduced to spectra. This is done using two different manners, from the whole segmentation map and from an aperture.

In both reduction cases, we apply the Marz [24] conventions, to ease the analysis by the tool.

The objects datacubes are preprocessed to mask invalid pixels. These include pixels containing a negative flux value, as well as those containing a NaN. To this mask is added an identical mask for variance: negative variance values and NaNs are flagged.

Then, for the segmentation map reduction, all pixels contained within the object and not masked are summed along spatial axis. The same happens for the variance extension pixels. These operations are performed using numpy masked arrays sum along axis. The consequence is that at the end of the operation, a spectral pixel may be masked if there are no pixels with a value at its wavelength band. These are replaced by NaN.

For the aperture reduction, the segmentation mask is replaced by a circular aperture - of radius specified by the user through the `-aperture` option, defaulting to 5. The center of the disk is determined through the `RA` and `DEC` keywords from the header, determined as explained previously in this subsection.

All spectra are written individually with three extensions. The primary one contains the spectrum, and WCS information about wavelength and spatial coordinates, as well as `TRUST` being the maximal number of detection in the segmentation map. A second extension contains `VARIANCE`, while the third is a table with three columns (`RA`, `DEC`, and `NAME` being the object ID) and a single row of values. Segmentation spectra are placed in the folder “spectra” while apertures one are written to “apertures” folder.

Then, we run on those two folders a function to merge all spectra in a folder into a single file, to ease analysis by Marz. We obtain two files, `combined_spectra_seg.fits` and `combined_spectra_apertures.fits`, that can be inserted to Marz directly.

4.7 Marz automatic analysis

Marz [24] is an automatic template-matching application, programmed in JavaScript, to analyze spectra. The tool matches a template and a redshift to an input signal, and output them along a correlation indicator, Xcor. The higher Xcor is, the higher the match is. To fit the parameters, Marz smooths spectrum using the variance extension, is able to remove the continuum of a spectrum, and reduces the importance of ends of the signal in its calculations — as they are often places of high noise. Marz provides two modes. The first one is graphical, provide the user a first set of input parameters, and allows selecting parameters (template and redshift) manually, as shown on Figure 4.10.

The second mode is automatic : the tool returns the 5 best templates matches to each spectrum, for both aperture and segmentation files.

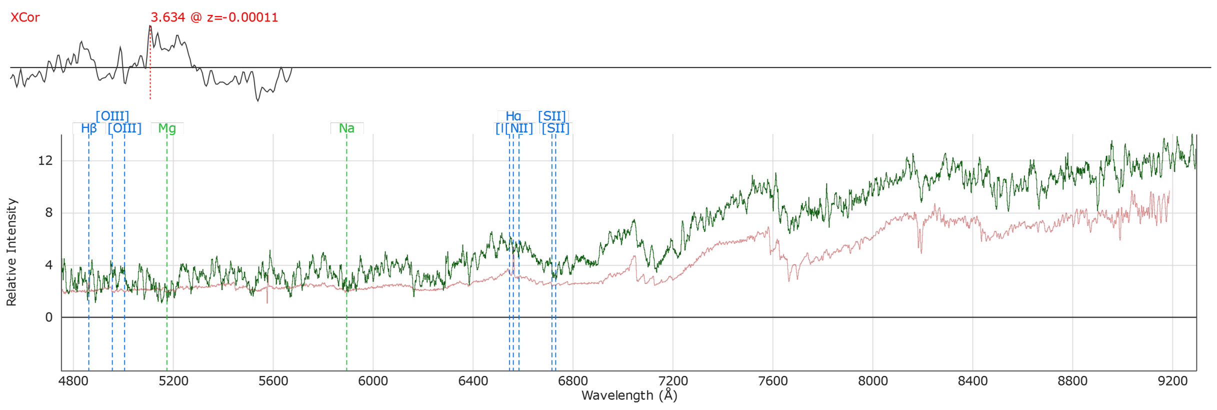

Figure 4.10: Example of a spectrum and of a template in Marz-Graphical interface.

Chapter 5

Results

To assert the results from AutoMUSE, we analyze three MUSE cubes: the Hubble Deep Field South [4], the Hubble Ultra Deep Field [5], and the quasar Q1422+23.

5.1 Asserting the results

In a first place, we gather all the objects extracted from each cube, and analyze them manually through a Jupyter notebook [30].

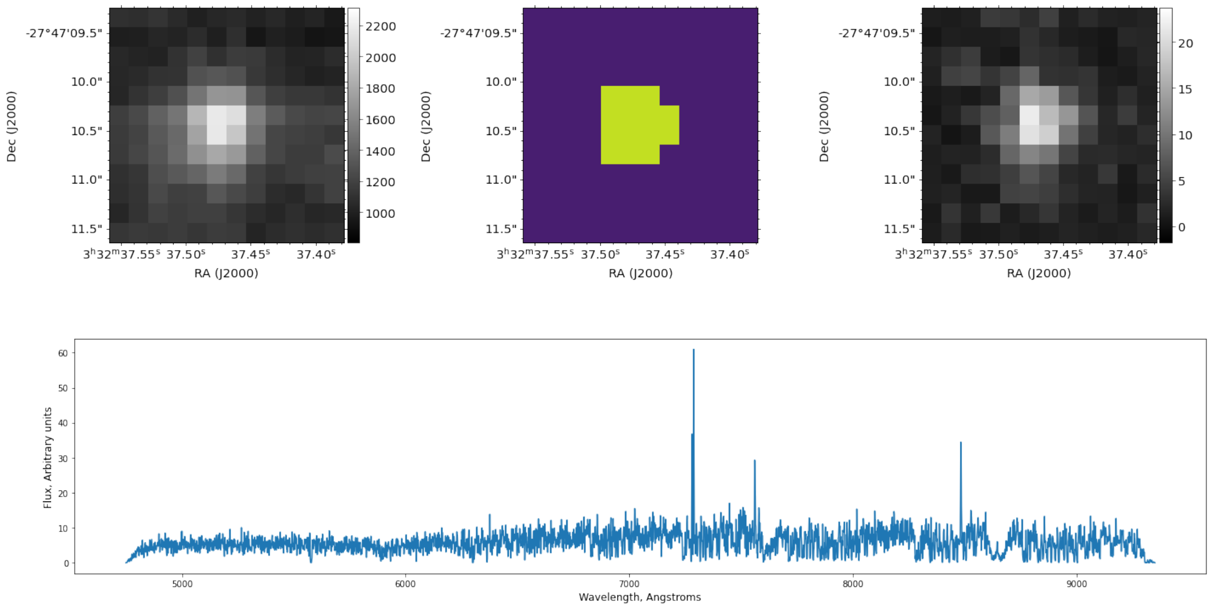

Figure 5.1: Hubble Ultra Deep Field (Figure 3.2) extracted object. Left panel: White-light image of the object, middle panel:Segmentation map of the object, right panel: narrow-band image of the object. Lower panel: Extracted 1D spectrum.

In the Jupyter Notebook, we plot each of the objects and their spectra as on Figure 5.1 . The upper left panel of Figure 5.1 is the white light image of the extracted object, the middle one represents its segmentation map. The right panel is the narrow-band image of the object. To extract it, we compute the spectrum (lower panel) similarly to the segmentation spectrum in section 4.6. We then fit the object continuum using [18] and normalize the flux to it and to its variance, similarly to Marz [24] :

$$S(\lambda) = \frac{\text{Flux}(\lambda)}{\text{fit}(\lambda) \times \sigma^2(\lambda)}$$

We search for the maximum of $S(\lambda)$, and extract the object's layers (not the spectrum) around that point, and compute the narrow-band image as in Figure 4.2. We also plot the spectrum (not variance and flux normalized, only cosine tapered) to check if there is an eye-recognizable pattern or if there is just noise. Along with each object plot, we search for three features. We check if the object is correctly fitted to its segmentation map, that is if the object is under-segmented or over-segmented. We verify its presence on the narrow-band, then on the white light image. Furthermore, we obtain a table, with 4 columns : object ID, object fitting, detection on the white-light image, and detection on the narrow band image.

5.1.1 Hubble Ultra-deep field: Detection results

On the Hubble Ultra Deep Field, with default parameters of AutoMUSE, we detect 593 objects. The segmentation map is available on Figure 3.3.

In these objects, we find the distribution of AutoMUSE findings exposed in Figure 5.2.

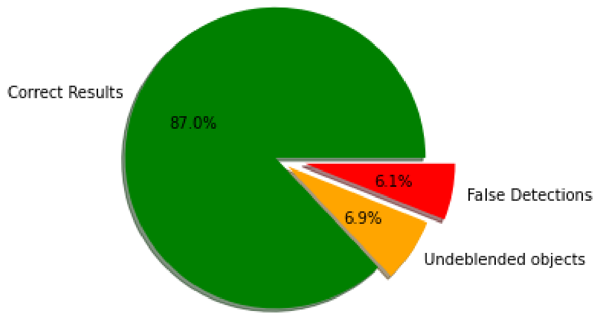

Figure 5.2: AutoMUSE findings in Figure 3.2. 516 objects appear to be correctly extracted and deblended. 41 objects are correct detections, but not correctly deblended: several sources are belonging to the same object. Finally, 36 are false detections, where no source clearly appear in broad and narrow-band images.

In these results, we find no case of over-segmented objects. However, some objects are undeblended, either in broad-band or narrow-band image. Illustrations of different situations leading to not deblended objects are available on Figure 5.3, Figure 5.4 and Figure 5.5, with their associated explanations.

The false detection are of two types : artifacts, as shown on Figure 5.6 and unsure objects, where a source could exist but is dominated by noise, Figure 5.7.

Finally, we achieve a high score (87%) in good — and deblended — detection, that spans a large variety of cases. A great part of the objects are detected on both white-light (broad band) and narrow-band images, but some are detected only on one of the two, as shown on Figure 5.8.

We encounter various situations within the correctly deblended objects. For example, on Figure 5.9, AutoMUSE successfully detects and deblend an emission-line source between two objects that are visible in the white-light image, completely masking it. On

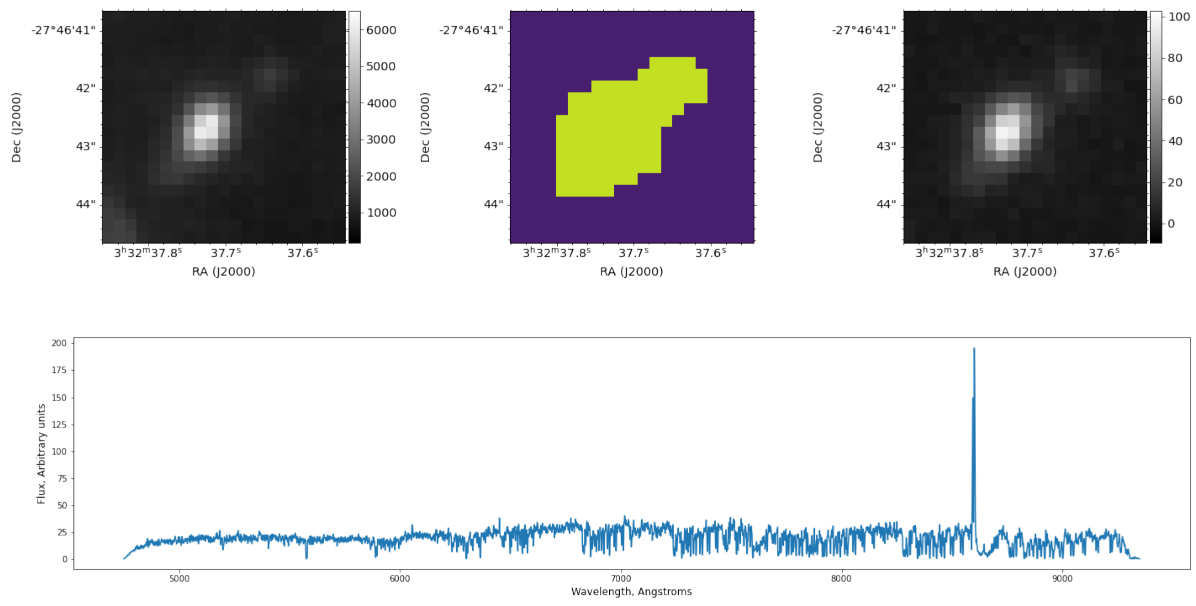

Figure 5.3: HUDF (Figure 3.2) object detected by AutoMUSE, but not correctly deblended. On the images we observe two objects, one much brighter than the second. This should have been deblended applying topological deblending from subsection 4.5.3, but this did not succeed on that object.

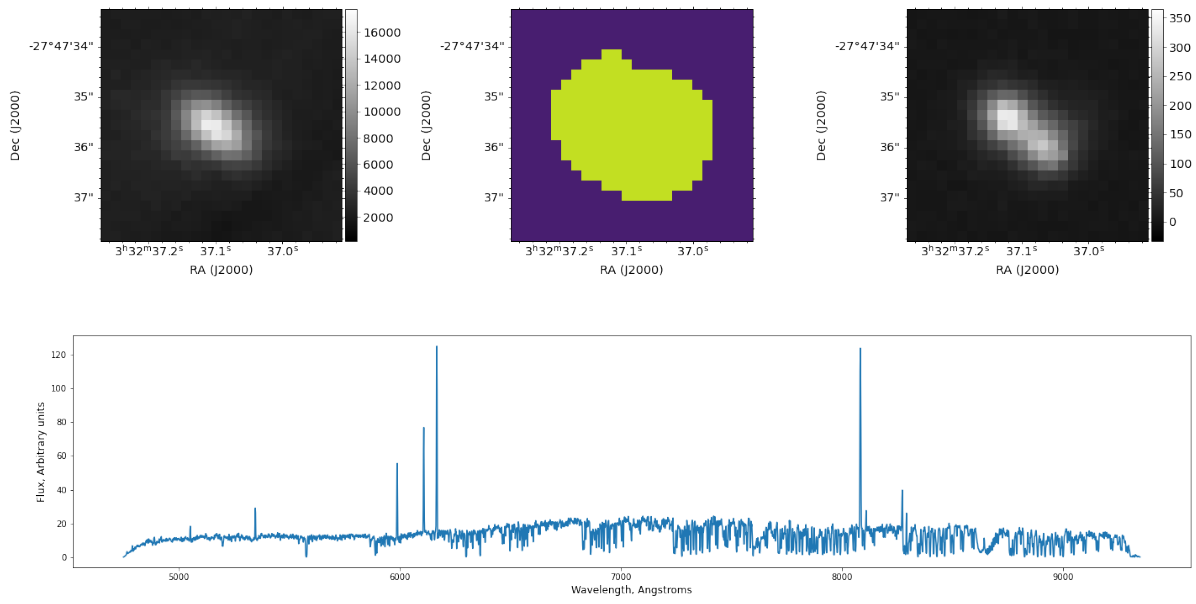

Figure 5.4: HUDF (Figure 3.2) object detected by AutoMUSE, but not correctly deblended. On the narrow-band image we observe two objects, while they appear unified in the white-light. This may be corrected by implementing a narrow-band deblending — see subsection 6.1.2.

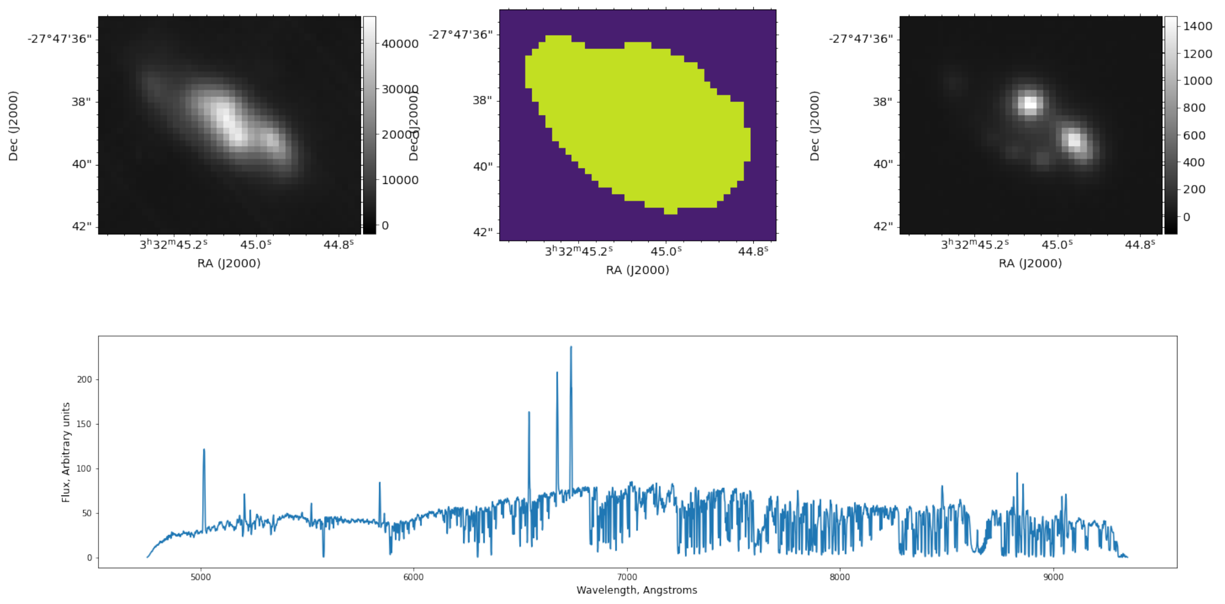

Figure 5.5: HUDF (Figure 3.2) object detected by AutoMUSE, but not correctly deblended. Looking carefully at it shows at least 5 objects. Starting with the narrow-band, one sees two bright and two fainter sources. Masking the region with these objects in the white-light, one can find the presence of a fifth source on the left-hand corner of the image - still in the segmentation map. This source is very faint, but guessable on the narrow band image. With additional masking, we found that a sixth source may be observable between the fifth object and the four others, without being certain if it could be an artifact.

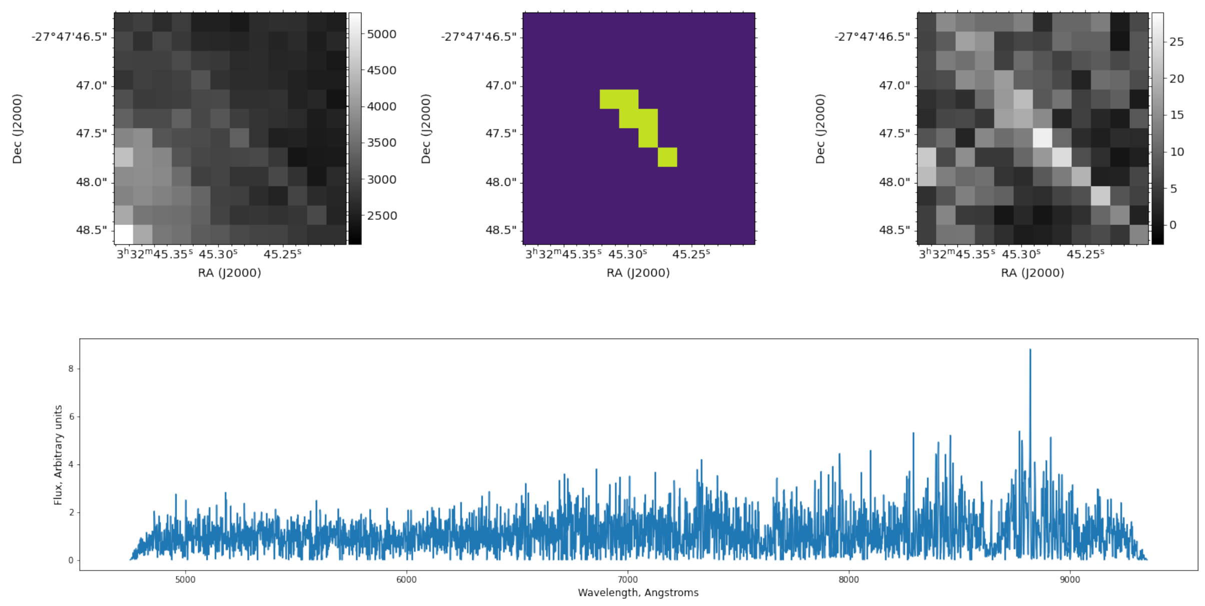

Figure 5.6: HUDF (Figure 3.2) false detection by AutoMUSE. This artifact is likely due to the method used to gather the ten cubes composing the field, as it lays on the frontier between two of them.

Figure 5.10, an emitter is successfully detected while merged (in the sky) with another object.

The object on Figure 5.11 is a single-emission line object blurred by a source in the upper left corner of the local cube, but is still detected by our script.

5.1.2 Q1422+23 field : Detection results

We perform the same operations as for the Hubble Ultra Deep Field on a smaller cube, containing the quasar target Q1422+23. This datacube is one of the cube that was used to test and calibrate the tool against very bright and extended objects. The quasar is a compound of bright gravitational lens - lensed object pair, composed of four sources ([51]). It has been observed by many instruments, including MUSE. We downloaded the object datacube identified as `MUSE.2015-04-24T04:20:38.757` from ESO archive. Its white-light on Figure 5.12. AutoMUSE produced a segmentation map on Figure 5.13, marking different sources in the field of view.

One of the challenges with AutoMUSE and the quasar was to prevent its over-segmentation, while being able to separate the four connected sources. The succession of deblending steps (section 4.5) tends to create a few regions in the area occupied by the quasi-stellar object, that are reunified by the spectral-fusing algorithm. The original SExtractor [9], if not fine-tuned, also had troubles to produce a correct map of the area. The manual classification of the 93 detected regions is available on Figure 5.14 an Figure 5.15.

As the reader can observe on Figure 5.14, the datacube containing the quasar is subject to a high false detection rate. This is due to the presence of sky emission lines in the field. Mainly, the $6297,9\text{Å}$ and the $OI6300\text{Å}$ sky lines [38] have not been removed properly from spectra. We learn from that cube that the sky lines must be properly cleaned in order to have accurate results.

As explained at the beginning of the section, AutoMUSE detected 593 sources (false detections included) in the Hubble Ultra Deep Field issued from MUSE, using a detection

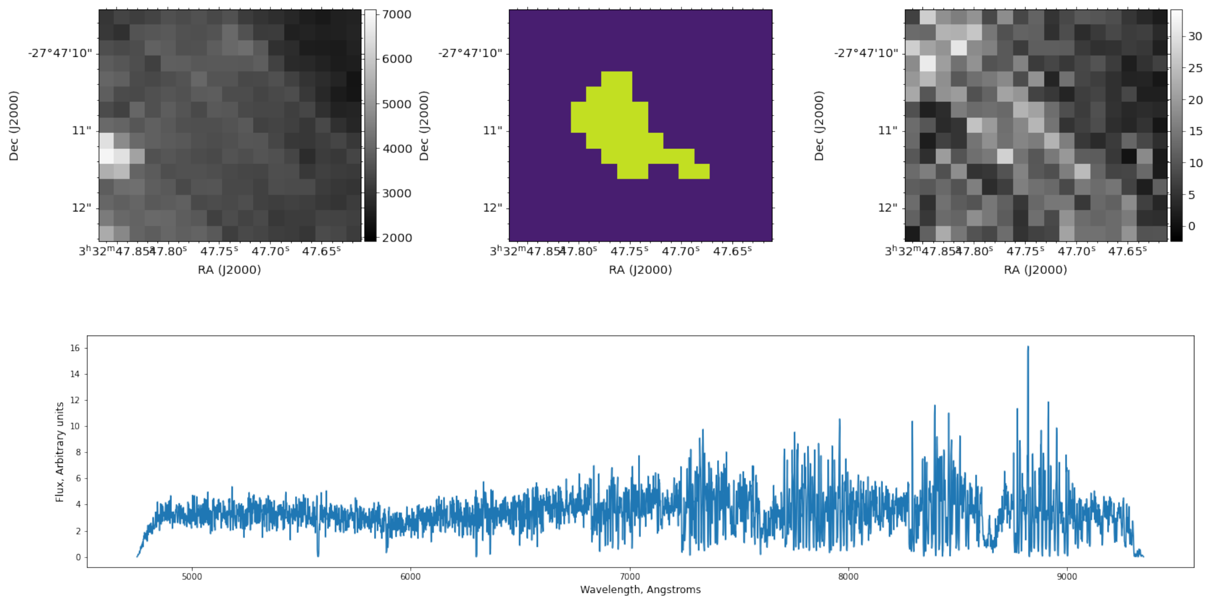

Figure 5.7: HUDF (Figure 3.2) probable false detection by AutoMUSE. We are not sure that an object is present in this region. The object is close to a frontier between two datacubes, and close to another object. However, there are clues of the presence of an object in the spectrum, that could be due to contamination from the nearby object.

threshold of 3σ. In [5], Bacon et al. detect 1251 "potential real detections" + 672 false detections using a different algorithm, ORIGIN [34]. We should justify that difference.

The two tools have a different philosophy and way of functioning. AutoMUSE, as explained thoroughly along this thesis, uses "standard" methods of detection, looking into broad and narrow-band images directions, computing backgrounds, etc. ORIGIN, on the contrary, looks in the spectral direction, and tries to match spectral signatures to the images, in order to detect emission lines. As shown in [5], ORIGIN performs well on high signal-to-noise parts of the mosaic compounding the HUDF, with a 86% purity of detections in the most exposed area of the cube. That ratio lowers down to 65% in regions with lower signal-to-noise. For a detection threshold $\sigma = 3$, AutoMUSE has a purity that goes up to 93.9% while including not deblended sources.

Finally, if we lower AutoMUSE detection threshold to $sigma = 2$, we detect 996 sources. If that number is lowered to $\sigma = 1.5$, 1373 objects are detected. Then, the same order-of magnitude of sources are detected using AutoMUSE with an appropriate threshold, compared to ORIGIN. However, we did not run a manual sanity check of these detections, so we can't affirm the purity of these detections.

5.1.3 Hubble Deep Field South

R. Bacon and al. [4] obtained a 3D view of the Hubble Deep Field South with 37 hours of observation of that region. We downloaded the version 1.34 of the cube on `muse-vlt.eu` website, as the data is in open access. For the HDFS target, we find the segmentation map Figure 5.17, that comport 203 different objects.

We see, in Hubble Deep Field South distribution figure, a lot of false detections. Objects images, as stated before, were inspected visually, without applying a special treatment to the spectra. Some objects, with a single very bright pixel (on the white-light or on the narrow-band), were considered as incorrect findings, due to our uncertainty on

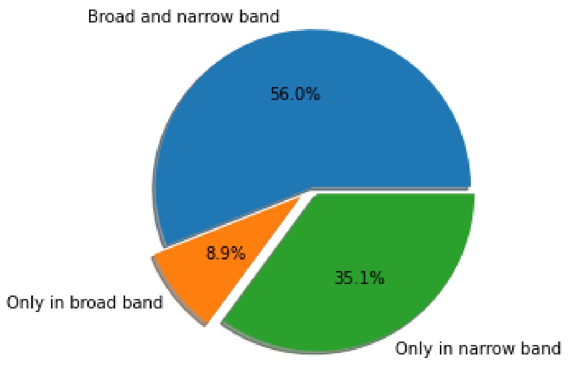

Figure 5.8: AutoMUSE correct results types in Figure 3.2. Among the 516 “correctly deblended objects” from the Hubble Ultra Deep Field, 289 are detected in both broad and narrow-band images, 181 in narrow-band only, and 56 in broad-band only.

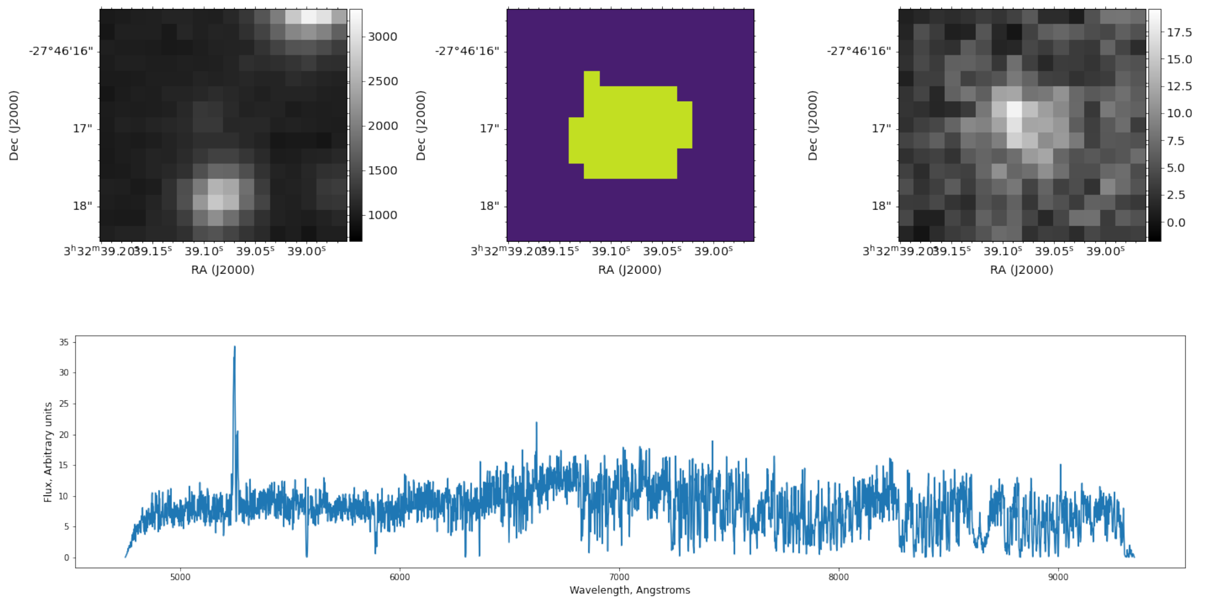

Figure 5.9: Figure 3.2 faint emission-line object between two brighter sources.

the presence of a source. To simplify the analysis, we did not include a “doubt” category to the pie chart, but note that on the 29 objects classified as ”False detection”, we had doubts on 12 of them. Analysis to confirm the presence of sources could be realized by looking at other instruments, looking in the optical range as MUSE does.

The Hubble Deep Field South has been analyzed by Bacon et al. [4], constructing a catalog of 585 objects available on their website[1].

In these objects, 189 have a “CONFIDENCE” marked as non-null. Within these objects, 44 are not detected in threshold 3 segmentation map issued by AutoMUSE. However, in the 203 objects of AutoMUSE, 79 are not found with a > 0 confidence in the HDFS catalog. While removing the confidence filter, 367 objects are not found on the segmentation map, and conversely, 26 are not in the catalog. If we remove from the segmentation map the 17 objects that are false detection, the number of AutoMUSE objects not found in the HDFS catalog with a > 0 confidence drops to 68.

[1]`http://muse-vlt.eu/science/hdfs-v1-0/`

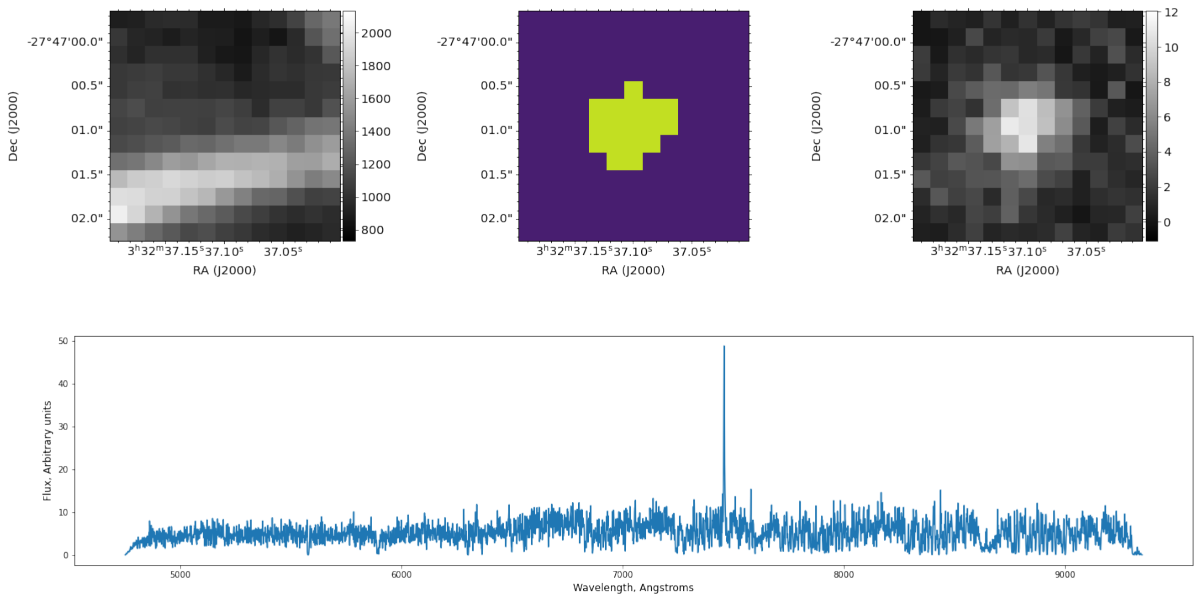

Figure 5.10: Figure 3.2 faint emission-line object hidden behind another extended source.

If we lower the threshold parameter of AutoMUSE to $\sigma = 1.5$, then we detect 272 objects. The number of objects not found in the segmentation map stays stable - 378. That (a bit) higher number can be explained by border effects: we remove objects touching edges of the image in AutoMUSE segmentation maps. However, 142 objects are not in the catalogs.

This data has to be taken carefully. The false detection have not been filtered in the $\sigma = 1.5$ map.

5.2 Spectral analysis of the Hubble Ultra Deep Field

To know more about the analyzed fields that have been used to prove the capabilities of AutoMUSE, we analyze the spectra issued from the segmentation map, using Marz.

As stated in section 4.7, Marz is a template matching tool, that associate a template and a redshift to a spectrum. In the Marz [24] version used, there are 22 templates : 5 of stars, 1 of a quasar and the others of galaxies of various types. These templates are eigenspectra, issued from multiple sources, as explained in the Marz article.

While in automatic classifier mode, Marz outputs a template+redshift match list, associated and sorted with a score called Xcor. Xcor is a function of the redshift based on the Fourier Transform of a spectrum and templates, and score returned are the highest local maxima. Marz also returns a "Quality Operator" (QOP) that is a function of the two greatest peaks of the Xcor function. The automatic QOP, that later needs to be confirmed manually while examining spectra, range from 1 to 6, where:

1 is Unknown/unsure.

2 is possible, there is a correlation between spectrum and templates+redshift combination, but is unsure.

3 is when there is a good correlation. In [24], the author classify that QOP to a 95% confidence of classification.

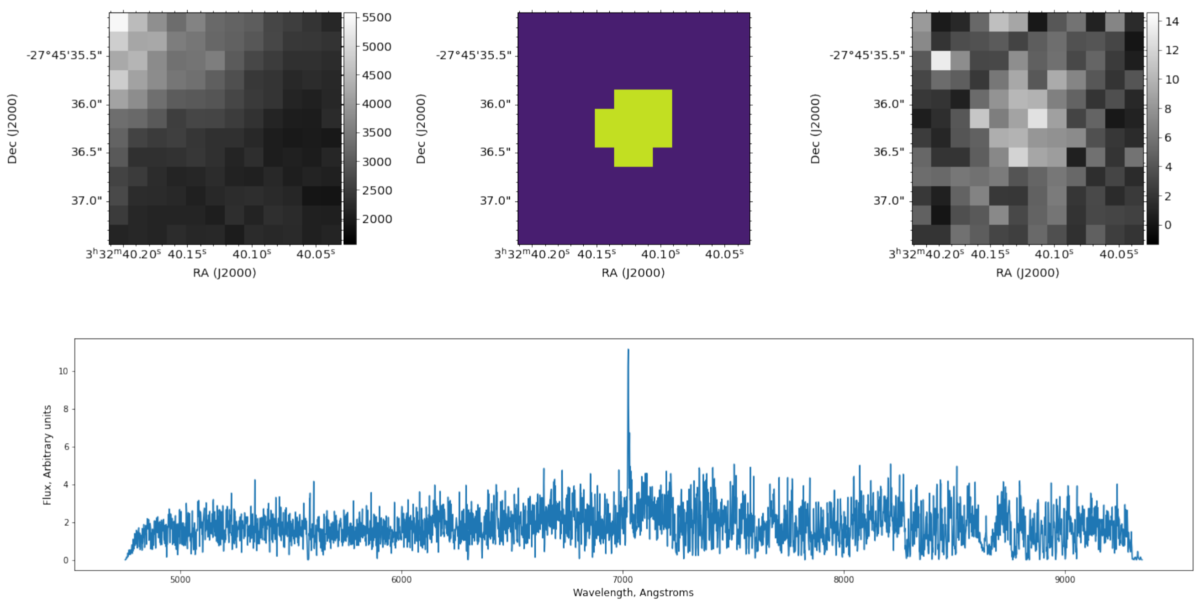

Figure 5.11: Hubble Ultra Deep Field (Figure 3.2) faint extracted object, even if close to a bright source.

4 represents a great correlation, with a 99.5% confidence.

5 : There is no QOP5 in Marz.

6 is dedicated to match stars. It is associated when the automatic QOP is >3 while matching a stellar template.

We visually analyze the spectra contained in the Hubble Ultra Deep Field, without knowing prior information on their deblending or misdetection state. We associate a manual QOP to each spectrum, and we keep the liberty to change parameters from the proposed templates+redshift combination automatically proposed by Marz. The manual QOP are defined according to visual inspection, with the following parameters:

1 is Unknown/unsure, or when the spectrum is too noisy to conclude.

2 is a possible resemblance between a template and a spectrum. Some emission/absorption lines match, but not the majority - or there are not enough lines on the spectrum to conclude. If there is a doubt between two templates, the QOP falls into that category.

3 is when there is a majority of lines that match, but not all of them. The template doesn't match "perfectly", or there is not enough lines to conclude but the spectrum match perfectly, without additional artifacts.

4 if almost all — if not all — lines (more than 5) match, and the template matches perfectly. If there is one or two lines that disagree, and they are not bright enough to be important, the match can fall into that category.

6 is dedicated to match stars, and used when the correlation is great, with a quality near to the one required falling in QOP 4.

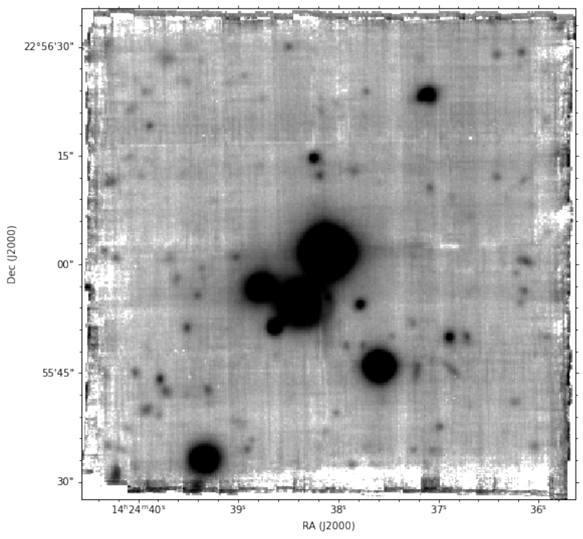

Figure 5.12: Q1422+23 segmentation map, by AutoMUSE.

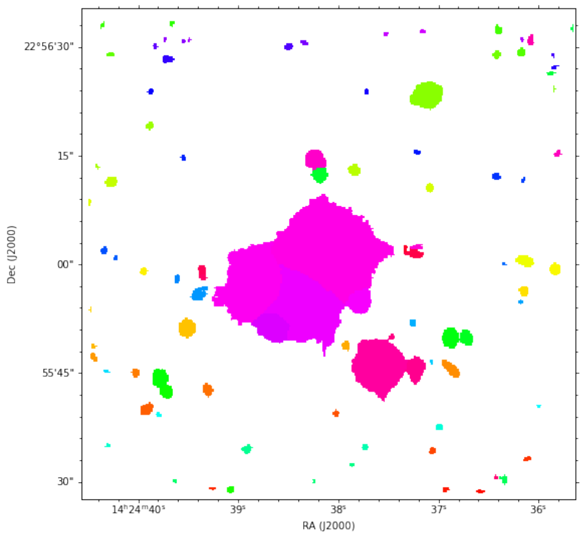

Figure 5.13: Segmentation map of Figure 5.12 issued by AutoMUSE.

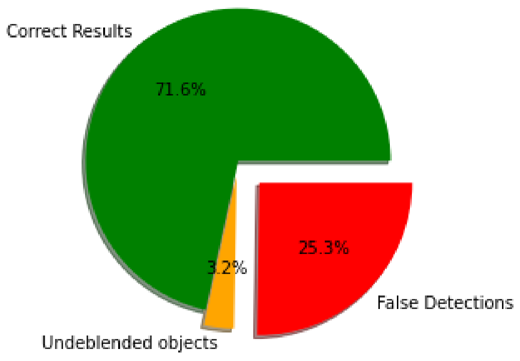

Figure 5.14: AutoMUSE findings in Q1422+23, Figure 5.12.

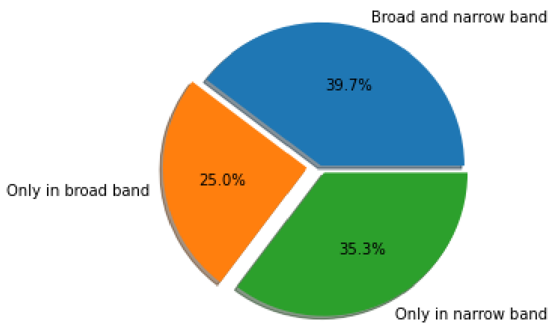

Figure 5.15: AutoMUSE correct results in Figure 5.12.

The results on manual QOP should be taken carefully. All spectra were inspected by a sole person, that is not an expert on template matching (ie, the writer of this thesis). Broad, faint lines can be unnoticed during the matching process. We take the image with the larger number of objects (Hubble Ultra Deep Field) to run this analysis.

On Figure 5.22, it is not surprising that we find that a majority of objects classified as false detection are QOP1, while the rest are QOP2. There is a single object in QOP3, which leads us to think that that object was faint, and put into the wrong category while looking at it by eye.

Comparing Figure 5.20 and Figure 5.21, we find that, proportionally, there are more QOP4 objects in the undeblended objects than in the deblended ones. However, in the correctly deblended objects, there are 92 QOP4, while they are just 19 in the undeblended ones.

Nevertheless, one can wonder why almost half of the undeblended objects are classified as QOP4, while several spectral signatures should mix into these sources. A clue, that we verified on three of the 19 undeblended QOP4, is that one of the sources in the segmentation map is much brighter than the others. Then, the emission is dominated by that bright spot, leading to the masking of the other sources in the spectrum.

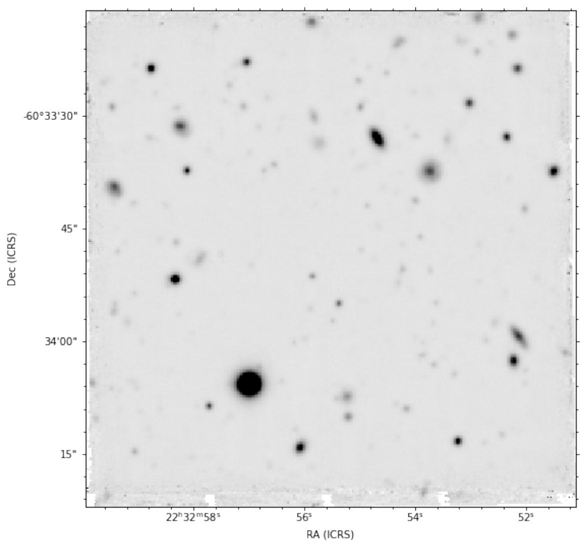

Figure 5.16: Hubble Deep Field South segmentation map, by AutoMUSE.

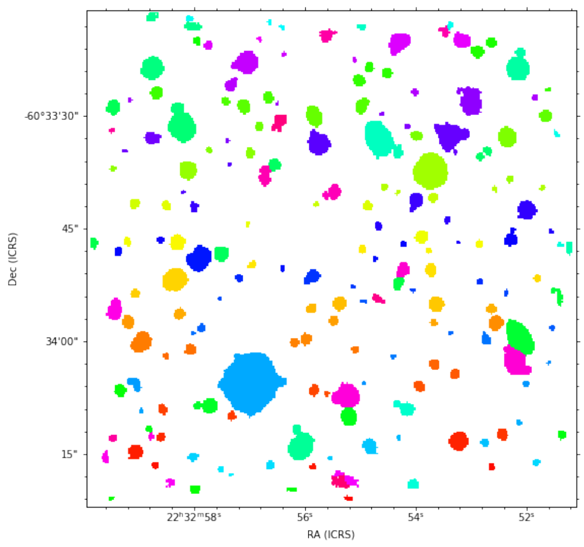

Figure 5.17: Segmentation map of Figure 5.16 issued by AutoMUSE.

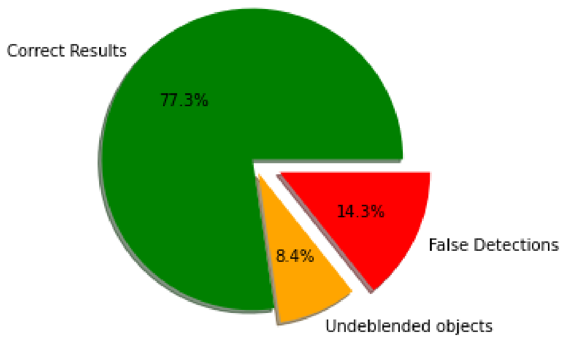

Figure 5.18: AutoMUSE findings in the Hubble Deep Field South, Figure 5.16.

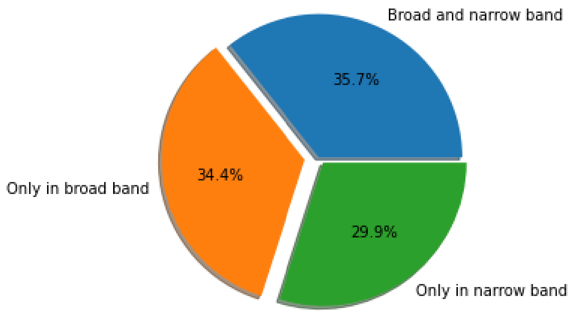

Figure 5.19: AutoMUSE correct results in Figure 5.12.

Finally, to explain a part of QOP2/3 on Figure 5.20, some sources are correctly deblended, but still contaminated by the light of nearby objects - that they were deblended from.

In an article about the MUSE Hubble Ultra Deep Field Survey [28], Inami et al explore the redshift of sources contained in the cube. A majority of them have a Hubble Space Telescope counterpart and were detected using external images from the space device, and some faint ones using ORIGIN and MUSELET.

From their redshift list, available as a Vizier catalog[2], we select the ones with a high confidence ("Confid" = 3). From them, we plot the histogram on Figure 5.23.

With our analysis, using spectra gathered by AutoMUSE, we obtain the histogram on Figure 5.24. We selected spectra from deblended objects, with QOP>=2.

[2]https://vizier.u-strasbg.fr/viz-bin/VizieR?-source=J/A%2BA/608/A2

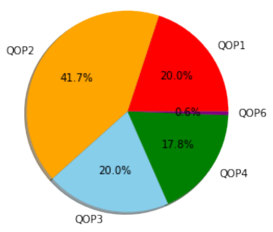

Figure 5.20: Marz QOP distribution among the objects marked as correctly deblended and detected in the Hubble Ultra Deep Field segmentation map.

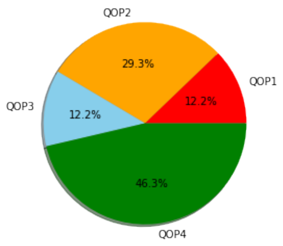

Figure 5.21: Marz QOP distribution among the objects marked as undeblended in the Hubble Ultra Deep Field segmentation map.

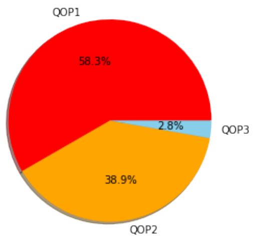

Figure 5.22: Marz QOP distribution among the objects marked as false detections in the Hubble Ultra Deep Field segmentation map.

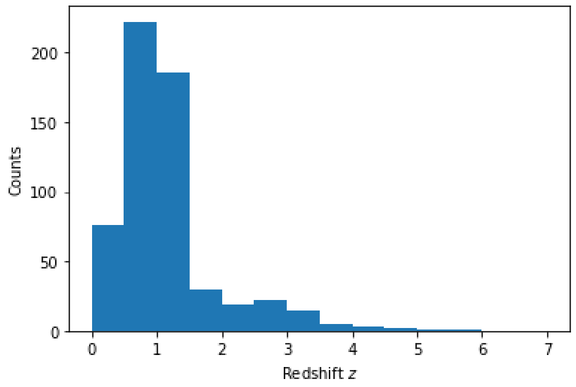

Figure 5.23: High-confidence spectroscopic redshift in MUSE Hubble Ultra Deep Field.

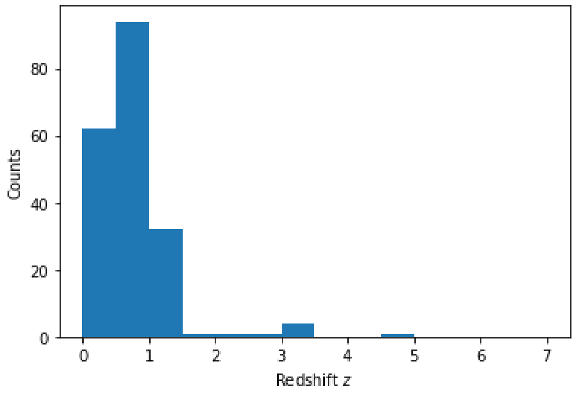

Figure 5.24: Spectroscopic redshift in MUSE Hubble Ultra Deep Field, by AutoMUSE detected sources.

Chapter 6

Challenges and conclusions

6.1 Challenges and potential developments

6.1.1 Preprocessing of datacubes

A source of issues in AutoMUSE is the data organization and preprocessing of fields. MUSE pipelines [49] performs well and create cubes that are readable by our program. However, these cubes can be modified or issued from another source, and WCS information can be lost in the process. On cubes we tested, the `BUNIT` keyword was sometimes removed, while being crucial for astropy nddata.CCDData interface. The WCS axis keywords (`CTYPEi`) are usually well-defined when describing spatial axis, but can be challenging when facing spectral definitions. When looking at that third dimension, `CTYPE` should be set to `AWAV` - Air WAVelength - or to `WAVE` — vacuum WAVElength, as explained in section 2.3. A filter is set in AutoMUSE to handle as accurately as possible various definitions, but we did not compile all of them.

Moreover, we started to implement in our tool an astrometric corrector. In order to produce more precise results on the position of detected objects, and to fix incorrect (or missing) header data, it could be an improvement to rely on additional sources than on the FITS file itself. We initially based that part of the work on `astrometry.net`, without success, due to the lack of stars in fields. We did not have enough time during this master thesis to create a function handling astrometry (a master thesis is just one year long), we removed that part from our code after exploring additional options, such as SCAMP [8]. If one has to run a source-detection algorithm on numerous MUSE files in archives, that part of software has to be implemented.

The final issue we encountered is due to data itself, and not headers. As shown in subsection 5.1.2, sky subtraction is of key importance, as it seems to be the highest source of false detection. The Zurich Atmosphere Pure, ZAP [44], is an important tool to remove sky residual, especially in MUSE products, and should probably be used on AutoMUSE cubes. A way we did not retained to reduce sky subtraction defects is to discard wavelength planes in which sky spectrum shows an emission line. Based on a sky emission line atlas - such as [38], we could disregard sources detection on a list of spectral coordinate, but we estimated that would be too much loss of information. Indeed, AutoMUSE has the capacity to detect both broad and emission line sources, and disregarding entire wavelength planes would reduce efficiency of detection of the latter.

6.1.2 AutoMUSE processing

We encountered many exceptions to handle while developing AutoMUSE. In theory, it should deal with all errors and scenarios encountered - absence of sources in an image, difficult background estimation for crowded fields or bad-pixels only in an object. However, we are aware that all exceptions were not considered, and we would be happy to correct errors a user may encounter.

AutoMUSE is fully based on `tqdm process_map` and `multiprocessing` to parallelize computations in terms of processes - and not threads. Processes do not interact with each other, and all execute the same task on different CPU cores. The key point of the algorithm is its deblending part. Through its different steps, described thoroughly long this thesis, we describe how we detect and segment objects. However, we show on Figure 5.5 that additional deblending steps could be added. For example, once an object has been extracted, one could analyze it further.

The first thing to check would be if there is a source in the extracted region. This could be done similarly to MUSELET, by checking a flux from the white light image.

We had some trails about an additional implementation filter, but lack of time to program them. In a first place, one can make a first detection pass, and gather object's spectra. Once that is done, the script could find two flux values : a first one by summing the entire spectrum — total flux — and one by obtaining the spectrum in a narrow window around the largest emission line — narrow flux. Fluxes would then be ordered, and the first quartile - or any other chosen quantile - of each list would be used, multiplied by a threshold (20%?), to flag objects with a lower flux value than that number in both lists. If an object sees its fluxes low enough to be flagged in both lists, it would be discarded.

Another filter can be added knowing limiting magnitude and flux, that are given by ESO on its website dedicated to MUSE instrument. As an example, in narrow field mode, the predicted limiting flux is $2.310^{-18} \mathrm{ergs}^{-1}\mathrm{cm}^{-2}$ per hour of observation. As the exposure time is available in FITS file headers, one could discard objects that are not brighter than the computed limiting flux.

In a second (additional) step, one can try to deblend objects again, case by case. One would need to decide some criteria that would start the new processing, such as the area occupied by the source segmentation map. On Figure 5.5, that step would be a deblending using a narrow band image, as one can clearly see by eye that numerous sources are present.

Finally, the last processing step of AutoMUSE would be on spectral reduction. Currently, to transform an object to spectra, we simply collapse segmentation maps and aperture masks of objects to the spectral axis, by a simple summation. One could improve spectral extraction by using techniques such as the ones presented in AutoSpec [21].

That Python tool, similarly to AutoMUSE, creates subcubes from a Large IFU datacube, based on a list provided by the user, while AutoMUSE handles the detection step. The tool then computes a cross-correlation map, between a reference 1D-spectrum, obtained by aperture or segmentation map summation, for example, and all spectra of the smaller cube. Then, only spectral pixels with a cross-correlation coefficient above a certain threshold would be added to the final 1D-spectrum, yielding, according to the AutoSpec authors, to a better S/N ratio. This technique is similar to the one used in subsection 4.5.6, where we use Pearson cross-correlation, not to discriminate pixels, but to prevent over-segmentation.

For testing purposes, we implemented the AutoSpec functions in AutoMUSE. We don't

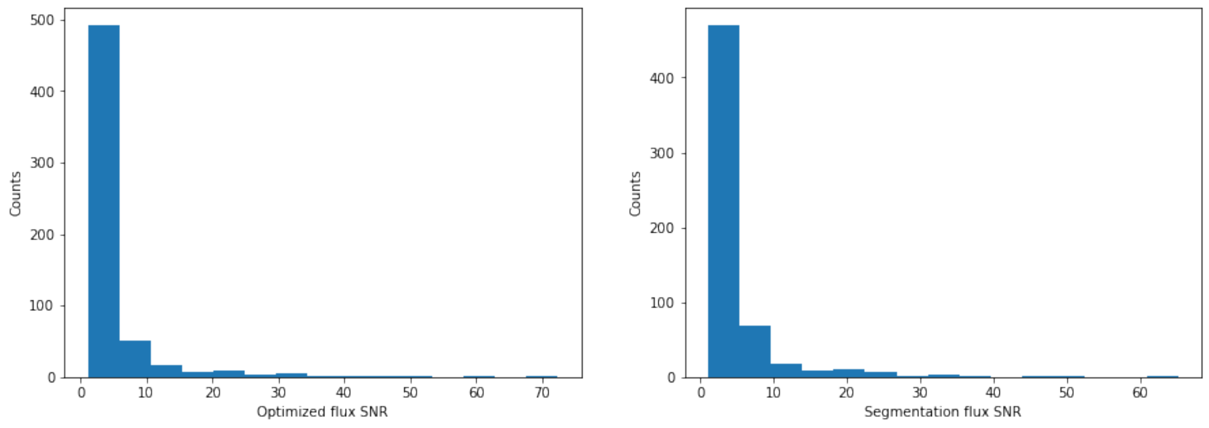

Figure 6.1: Det_SNR signal-to-noise of HUDF spectra

guarantee an identical functioning to AutoSpec. The reduction to "optimal" spectra can be enabled by setting the `-optimal` parameter to "True".

In that case, we realize the operations described in AutoSpec [21] section 2.2. In that section, Griffiths et al. subtract continuum from all spectra - in the subcube containing the source, and in the reference spectrum. Then they obtain the cross-correlation map (and the optimal spectrum) using spectral lines only. In AutoMUSE, reference spectrum is the one obtained from the segmentation map. The continuum is fitted (for each spectra in the object subcube) using a 8-degree polynomial. Then, we compute the cross-correlation map of the datacube, and extract the correlation pixels that are above $\sigma = 1.5$ (that's the default SExtractor parameter). We use that new segmentation map to extract a new reference spectrum - and repeat the operation a second time. The final cross-correlation segmentation map is used to extract a 1D optimal spectrum. 1D optimal spectra are written to the disk, similarly to aperture and segmentation spectra.

We computed segmentation and optimal spectra for the 593 sources in the Hubble Ultra Deep Field. We apply the DER_SNR algorithm [33] to extract a signal-to-noise ratio of both extracted 1D-spectra, and obtained the SNR distribution in Figure 6.1.

These distributions are quite similar. However, they contain a major flaw: the optimal flux extractor algorithm default to the segmentation map spectrum in case of any error.

Thus, we can compute the ratio between both SNR, optimal SNR against segmentation one. We remove the points where that ratio is equal to 1, and plot the distribution in Figure 6.2.

The distribution is slightly tilted in favor of the segmentation map extraction.

Nevertheless, we can find an use to the optimally extracted spectrum, and in particular of their cross-correlation map. In Figure 6.3, one can see that, when the segmentation-map area becomes large enough, the SNR ratio is often larger than 1. Actually, the cross-correlation map reduced the size of the object segmentation, and acted as a deblending technique, such as for the object in Figure 5.5. You can see it's cross-correlation map in Figure 6.4.

There is a little step to program from this cross-correlation technique to further refine the deblending process of AutoMUSE.

These techniques would provide a better signal-to-noise ratio. Additionally, one may need to clean contaminants from nearby sources.

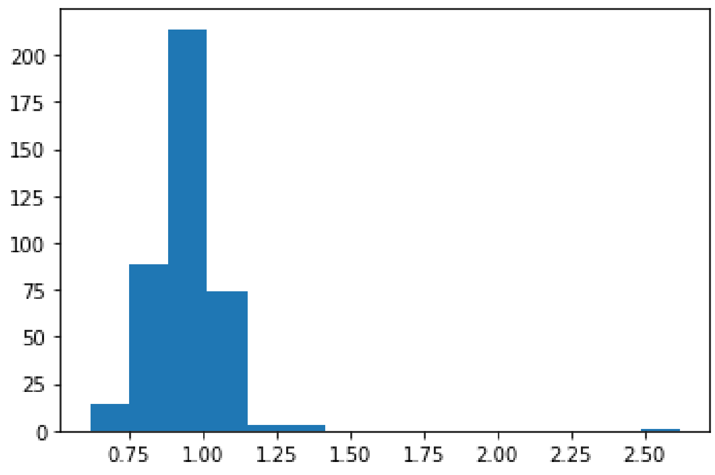

Figure 6.2: Ratio of Det_SNR signal-to-noise from HUDF spectra distribution. The mean is $\mu = 0.95$, median $m = 0.96$ and standard deviation $\sigma = 0.13$

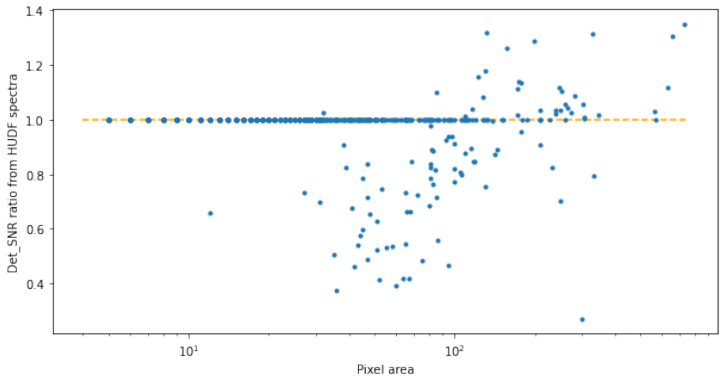

Figure 6.3: Ratio of Det_SNR signal-to-noise from HUDF spectra (y-axis) against pixel area occupied by the segmentation map (x-axis). Orange dashed-line at ratio=1.

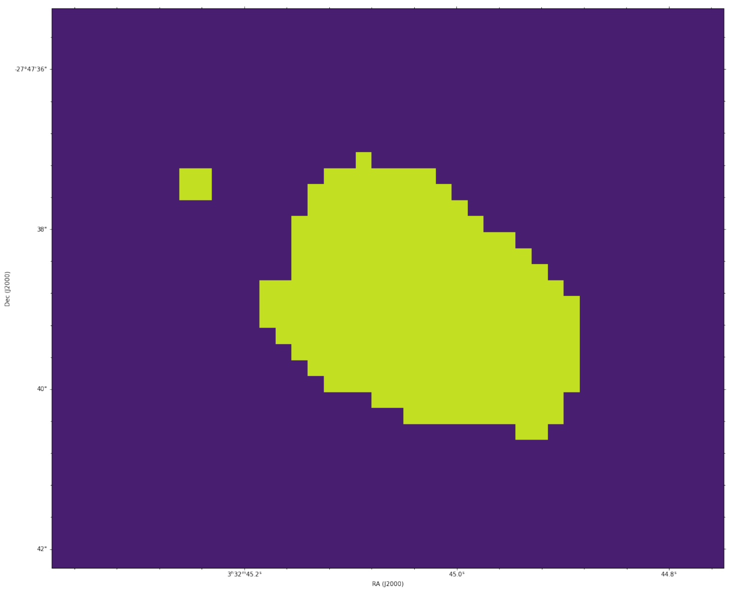

Figure 6.4: Cross correlation map of the object in Figure 5.5. The σ-threshold was set to 10.

6.1.3 AutoMUSE post-processing: An incursion into automatic classification

The original title of our MSc thesis was “Automatic detection and classification of MUSE spectra”. The automatic classification part was supposed to be managed entirely by Marz, as we thought we would be able to discriminate redshift automatically, based on the XCor parameter issued by the tool. In the current version of the code, available on Github[1], there is an attempt of automatic classification, that does not give better results than selecting the highest XCor template Marz select.

As explained earlier, AutoMUSE outputs two spectra per detected source, one from the segmentation map, the other from an aperture centered around the center-of-light of the object. Marz analyzes them automatically and output 5 template per spectrum, that is ten templates per objects. We combine the templates from the two analysis through a set of rules : if two templates have an identical name and a close redshift between the segmentation and apertures list of matches, they are added together and we artificially increase the XCor of the match.

That sum intervene so that two high-quality matches substantially increase the XCor, while two medium-to-low Xcor do not false the correct result. In order to perform such a sum, we chose to use the following form for the “sum”:

$$\text{XCor} = \log\left(e^{\text{XCor}_1} + e^{\text{XCor}_2}\right)$$

However, this should be taken carefully, as it doesn’t respect identity properties of a classic sum. It was only a way to combine and analyze two lists of templates.

The next operation was to boost Xcor for specific templates, that are stars. If a star is detected, then we boost the Xcor according to its redshift, if it ranges in the typical star velocity field to hyper-velocity stars. So, if a star template is associated to a (absolute) redshift lower than has a redshift associated to 400km/s, its XCor is boosted by 2, and if it is lower than 1000km/s, it is boosted by one.

Then, we consider that the highest final XCor is the correct template. We tested that algorithm on the objects issued from the Hubble Ultra Deep Field (subsection 5.1.1), for objects with a (manual) Marz QOP of 3 and 4 (223 objects). We chose to apply a tolerance of 0.002 for the redshift (two redshifts are considered identical if their difference is less than 0.002).

Marz is correct facing the manual z for 176 objects (that is a ratio of 0.79%). Our algorithm improves that ratio of 2 objects. The results are then not good enough to be implemented, as it may be source of additional errors. We then turned to potentially more efficient algorithms.

Machine learning is a branch of computational algorithms in which programs “learn by themselves” against a set of rules. In that family, there are many fields growing, like neural networks, that tend to emulate the functioning of human brain.

To our classification system, we tried to provide a tool, that would automatically predict a redshift from different Marz templates match. This operation is a regression in the frame of Machine learning. In order to do so, we downloaded spectra from the SDSS data release 16 [1]. From these, we selected the ones with no warning associated - meaning the redshift determined by the SDSS software is quite secure. We ran Marz on the downloaded spectra, and obtained Marz results for ≈500 000 sources.

[1]`https://github.com/Tenivar/automuse`

We randomly split the results into groups of 5000, and compare the Marz first template automatic redshift to the SDSS ones, with the formula:

$$\text{abs}\left(z_{\text{Marz}} - z_{\text{SDSS}}\right) < 3 \times \sigma_{\text{SDSS}}$$

where σ_{SDSS} is the error associated to a SDSS spectrum. Doing so, we find that $63.0 \pm 0.6\%$ of the time, Marz is correct against SDSS redshift. Additionally, if we look at $5 - \sigma$ SDSS errors, we find that that number grows to $76.3 \pm 0.5\%$. We also computed that in 85% of the cases (using SDSS spectra), one of the redshifts proposed in the 5 templates is within 5σ of SDSS redshift. Thus, we have to do better with machine learning! We used the scikit-learn [39] library in order to process computations.

We would like to note that the author of this thesis is a beginner in the machine learning field. We just wanted to try that technology, in the framework of a master thesis, as the redshift regression was one of the first goals of the thesis. To train the algorithm, we used the following features:

redshifts The 5 redshift proposals issued by Marz,

Automatic QOP The 5 automatic QOP issued by Marz, reorganized to range from 0 to 1 (so QOP 1 is 0.25, QOP 2 is 0.5 ...)

XCor The 5 XCor parameters issued by Marz

Template types for each of the 5 templates issued by Marz, a one-hot vector indicating if the template is from a quasar ([1 0 0]), a galaxy ([0 1 0]), or a star ([0 0 1]).

Position and flux of the strongest emission line.

We chose to use the decision tree regressor algorithm provided by scikit-learn.This choice was motivated by the fact we would like the algorithm to select a correct template among the various proposal issued by Marz. We transform the redshift distribution with the `QuantileTransformer` proposed, with a normal output distribution.

Once done, we fit the decision tree a hundred times, with different training and testing sets. We measure various scores:

Max Error measures the maximum error between prediction and target values. It has to be minimized;

explained variance is defined as

$$1 - \frac{\text{Var}\{y - \hat{y}\}}{\text{Var}\{y\}}$$

. The highest score achievable with explained variance is 1, and lower values means that predicted values are getting away from target values.

Accuracy computes the proportion of predicted results falling within 5σ of the SDSS redshifts.

We start with a maximum depth of the tree of 5. We find a max error of 6.84 ± 0.06, an explained variance of 0.46 ± 0.01, and an accuracy of 0.12 ± 0.01. After that, we decided to repeat the training until we hit a maximum in accuracy. With a maximum depth of the tree of 60, we obtain a max error of 6.97 ± 0.02, an explained variance of 0.44 ± 0.01, and an accuracy of 0.76 ± 0.01.

Thus, our decision tree classifier did not fit the parameters correctly to obtain a higher accuracy, of the order of 85% as we targeted.

6.2 Conclusions

We produced a Python 3 script, based on multiple Python 3 science libraries, that can generate segmentation maps from MUSE datacubes. Before using AutoMUSE on a cube, the latter should be formatted as close as possible to the astropy nddata.CCDData structure, with data and statistic extensions. As shown in subsection 5.1.2, IFU cubes need to be preprocessed to remove sky effects from observations, in order to limit false detection-rate. These cubes should also contain accurate and standardized World Coordinate System.

Once AutoMUSE is used on a datacube, it produces files accross its different steps. Narrow and broad band images are first used to create undeblended source maps, that are summed to a detection image. That detection image, qualifying how many times a pixel is declared as a source accross broad and narrow-band images, is used to deblend objects from connected regions, through a watershed algorithm. That watershed algorithm is used two additional times, on a topographic map of objects and on the white light image of sources. AutoMUSE implements two mechanisms to limit over-segmentation. In a first place, once an object is flagged as "elliptical enough", it is not analyzed further. After deblending steps, the script also compare spectra of touching sources, and fuses the objects that have a high Pearson Correlation Coefficient between their spectra, thus limiting over-segmentation with a second algorithm.

Once the final segmentation map has been produced, along with a "trust" image (that is the ratio between the "detection image" and the number of broad and narrow-band images), objects are extracted and put in different files individually. They can be easily read with the astropy nddata.CCDData interface, and handle data, uncertainty (usually, variance) and mask (opposite of the segmentation of the object) FITS extensions. These extracted objects are summed along spatial axis to produce spectra files. These files are then gathered into two files : one containing spectra issued from segmentation maps, the second containing spectra issued from a circular aperture sum.

Finally, Marz is used to automatically assign 5 sets of redshifts and templates to each spectrum, permitting to draw some automatic statistics on the analyzed datacube.

Using AutoMUSE, we detected and extracted continuum and line-emission sources in a standardized way, from different fields. We then manually analyzed them, in order to examine the quality of the program, and to give a redshift distribution of sources in IFU products.

The goal of this thesis was to produce a program that could, without manual intervention, detect and classify sources from datacubes. This goal is explained by the great amount of MUSE products available in the ESO archives. Indeed, MUSE is often used to capture an object spectrum, without analyzing the other sources that are present around the original target. We would like to run our script on archives, in order to produce catalogs from these disregarded sources.

Another application is about quasar gravitational lensing, and more generally gravitational lensing. Being able to locate, characterize, and study sources involved or around a lensing event can be challenging. But to understand more clearly the behavior of lenses and lensed objects, astronomers need a tool to catalog surrounding objects, in order to verify the involvement of these in lens models.

Currently, we don't judge AutoMUSE mature enough to run on an archive without a user assisting it.

However, it can be used to produce segmentation maps, in a standardized way, and to extract objects and spectra automatically. It is easily modifiable, and one can add and adjust functions to detect, deblend and extract objects without technical difficulties, except from knowing Python 3. The code provided in Appendix B permits to assess results from extracted objects.

Other tools, such as MUSELET and ORIGIN are dedicated to blindly detect sources in IFU images, using different techniques than the ones developed in this MSc thesis. Similarly, they come with false detections, and as explained in the results section of this thesis, there must be a manual check of the results to assess their purity. These techniques can be complementary, in particular when using ORIGIN and AutoMUSE. Indeed, both algorithms look to different scopes : spectral signatures (for ORIGIN) and images (for AutoMUSE).

In both MUSELET and ORIGIN, documentations emphasize the fact that parameters must be selected carefully, step by step. In AutoMUSE, their is a single parameter to change, `-threshold`. Number of CPU and narrow-band spectral pixels size are modifiable, but should work by default. Thus, our program is easier to use than others.

Bibliography

[1] Romina Ahumada et al. "The 16th Data Release of the Sloan Digital Sky Surveys: First Release from the APOGEE-2 Southern Survey and Full Release of eBOSS Spectra". In: *The Astrophysical Journal Supplement Series* 249.1 (June 25, 2020), p. 3. ISSN: 0067-0049. DOI: 10.3847/1538-4365/ab929e. URL: https://iopscience.iop.org/article/10.3847/1538-4365/ab929e/meta (visited on 04/30/2021).

[2] Astropy Collaboration et al. "Astropy: A Community Python Package for Astronomy". In: *Astronomy and Astrophysics* 558 (Oct. 1, 2013), A33. ISSN: 0004-6361. DOI: 10.1051/0004-6361/201322068. URL: http://adsabs.harvard.edu/abs/2013A%26A...558A..33A (visited on 03/12/2021).

[3] Astropy Collaboration et al. "The Astropy Project: Building an Open-Science Project and Status of the v2.0 Core Package". In: *The Astronomical Journal* 156 (Sept. 1, 2018), p. 123. ISSN: 0004-6256. DOI: 10.3847/1538-3881/aabc4f. URL: http://adsabs.harvard.edu/abs/2018AJ....156..123A (visited on 03/12/2021).

[4] R. Bacon et al. "The MUSE 3D View of the Hubble Deep Field South". In: *Astronomy & Astrophysics* 575 (Mar. 1, 2015), A75. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361/201425419. URL: https://www.aanda.org/articles/aa/abs/2015/03/aa25419-14/aa25419-14.html (visited on 03/19/2021).

[5] Roland Bacon et al. "The MUSE Hubble Ultra Deep Field Survey - I. Survey Description, Data Reduction, and Source Detection". In: *Astronomy & Astrophysics* 608 (Dec. 1, 2017), A1. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361/201730833. URL: https://www.aanda.org/articles/aa/abs/2017/12/aa30833-17/aa30833-17.html (visited on 05/09/2021).

[6] I. K. Baldry. "What Hubble Really Meant by Late and Early Type: Simply More or Less Complex in Appearance". Version 1. In: *Astronomy & Geophysics* 49.5 (Oct. 2008), pp. 5.25–5.26. ISSN: 13668781, 14684004. DOI: 10.1111/j.1468-4004.2008.49525.x. arXiv: 0809.0125. URL: http://arxiv.org/abs/0809.0125 (visited on 02/05/2021).

[7] Kyle Barbary. "SEP: Source Extractor as a Library". In: *Journal of Open Source Software* 1.6 (Oct. 5, 2016), p. 58. ISSN: 2475-9066. DOI: 10.21105/joss.00058. URL: https://joss.theoj.org/papers/10.21105/joss.00058 (visited on 03/12/2021).

[8] E. Bertin. "Automatic Astrometric and Photometric Calibration with SCAMP". In: 351 (July 1, 2006), p. 112. URL: http://adsabs.harvard.edu/abs/2006ASPC..351..112B (visited on 04/27/2021).

[9] E. Bertin and S. Arnouts. “SExtractor: Software for Source Extraction.” In: *Astronomy and Astrophysics Supplement Series* 117 (June 1, 1996), pp. 393–404. ISSN: 0365-0138. DOI: 10.1051/aas:1996164. URL: http://adsabs.harvard.edu/abs/1996A%26AS..117..393B (visited on 03/11/2021).

[10] R. B. Blackman and J. W. Tukey. “The Measurement of Power Spectra from the Point of View of Communications Engineering — Part I”. In: *Bell System Technical Journal* 37.1 (1958), pp. 185–282. ISSN: 1538-7305. DOI: 10.1002/j.1538-7305.1958.tb03874.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1958.tb03874.x (visited on 04/01/2021).

[11] Adam S. Bolton et al. “Spectral Classification and Redshift Measurement for the SDSS-III Baryon Oscillation Spectroscopic Survey”. In: *The Astronomical Journal* 144 (Nov. 1, 2012), p. 144. ISSN: 0004-6256. DOI: 10.1088/0004-6256/144/5/144. URL: http://adsabs.harvard.edu/abs/2012AJ....144..144B (visited on 05/07/2021).

[12] Larry Bradley et al. *Astropy/Photutils: 1.0.2*. Zenodo, Jan. 20, 2021. DOI: 10.5281/zenodo.4453725. URL: https://zenodo.org/record/4453725 (visited on 03/12/2021).

[13] Véronique Buat. *Lecture2.Pdf*. URL: https://people.lam.fr/buat.veronique/Veronique/Teaching_files/Lecture2.pdf (visited on 03/04/2021).

[14] Ronald J. Buta. “Galaxy Morphology”. In: *arXiv e-prints* 1102 (Feb. 1, 2011), arXiv:1102.0550. URL: http://adsabs.harvard.edu/abs/2011arXiv1102.0550B (visited on 03/03/2021).

[15] M. R. Calabretta and E. W. Greisen. “Representations of Celestial Coordinates in FITS”. In: *Astronomy & Astrophysics* 395.3 (Dec. 2002), pp. 1077–1122. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361:20021327. URL: http://www.aanda.org/10.1051/0004-6361:20021327 (visited on 05/09/2021).

[16] D. L. Crawford. “Light Pollution, an Environmental Problem for Astronomy and for Mankind”. In: *Memorie della Societa Astronomica Italiana* 71 (2000), p. 11. ISSN: 0037-8720. URL: https://ui.adsabs.harvard.edu/abs/2000MmSAI..71...11C/abstract (visited on 03/01/2021).

[17] Gerard de Vaucouleurs. “Integrated Colors of Bright Galaxies in the u, b, V System.” In: *The Astrophysical Journal Supplement Series* 5 (Jan. 1961), p. 233. ISSN: 0067-0049, 1538-4365. DOI: 10.1086/190056. URL: http://adsabs.harvard.edu/doi/10.1086/190056 (visited on 03/03/2021).

[18] Nicholas Earl et al. *Astropy/Specutils: V1.2*. Zenodo, Mar. 14, 2021. DOI: 10.5281/zenodo.4603801. URL: https://zenodo.org/record/4603801 (visited on 04/09/2021).

[19] George Forbes. *History of Astronomy*. Literary Licensing, LLC, Aug. 7, 2014. 220 pp. ISBN: 978-1-4981-4757-6.

[20] Hernan Grecco. *Hgrecco/Pint*. Mar. 11, 2021. URL: https://github.com/hgrecco/pint (visited on 03/12/2021).

[21] Alex Griffiths and Christopher J. Conselice. “AUTOSPEC: Fast Automated Spectral Extraction Software for IFU Data Cubes”. In: *The Astrophysical Journal* 869 (Dec. 1, 2018), p. 68. ISSN: 0004-637X. DOI: 10.3847/1538-4357/aaee87. URL: http://adsabs.harvard.edu/abs/2018ApJ...869...68G (visited on 04/29/2021).

[22] Olivier R. Hainaut and Andrew P. Williams. “Impact of Satellite Constellations on Astronomical Observations with ESO Telescopes in the Visible and Infrared Domains”. In: *Astronomy & Astrophysics* 636 (Apr. 1, 2020), A121. ISSN: 0004-6361, 1432-0746. DOI: `10.1051/0004-6361/202037501`. URL: `https://www.aanda.org/articles/aa/abs/2020/04/aa37501-20/aa37501-20.html` (visited on 02/23/2021).

[23] Charles R. Harris et al. “Array Programming with NumPy”. In: *Nature* 585.7825 (7825 Sept. 2020), pp. 357–362. ISSN: 1476-4687. DOI: `10.1038/s41586-020-2649-2`. URL: `https://www.nature.com/articles/s41586-020-2649-2` (visited on 03/12/2021).

[24] S. R. Hinton et al. “Marz: Manual and Automatic Redshifting Software”. In: *Astronomy and Computing* 15 (Apr. 1, 2016), pp. 61–71. ISSN: 2213-1337. DOI: `10.1016/j.ascom.2016.03.001`. URL: `https://www.sciencedirect.com/science/article/pii/S2213133716300166` (visited on 03/04/2021).

[25] B. W. Holwerda. *Source Extractor for Dummies V5*. Dec. 6, 2005. arXiv: `astro-ph/0512139`. URL: `http://arxiv.org/abs/astro-ph/0512139` (visited on 03/11/2021).

[26] Ben Hoyle et al. “Feature Importance for Machine Learning Redshifts Applied to SDSS Galaxies”. In: *Monthly Notices of the Royal Astronomical Society* 449.2 (May 11, 2015), pp. 1275–1283. ISSN: 0035-8711. DOI: `10.1093/mnras/stv373`. URL: `https://doi.org/10.1093/mnras/stv373` (visited on 05/07/2021).

[27] E. P. Hubble. “Extragalactic Nebulae.” In: *The Astrophysical Journal* 64 (Dec. 1926), pp. 321–369. ISSN: 0004-637X. DOI: `10.1086/143018`. URL: `https://ui.adsabs.harvard.edu/abs/1926ApJ....64..321H/abstract` (visited on 03/03/2021).

[28] H. Inami et al. “The MUSE Hubble Ultra Deep Field Survey. II. Spectroscopic Redshifts and Comparisons to Color Selections of High-Redshift Galaxies”. In: *Astronomy and Astrophysics* 608 (Dec. 1, 2017), A2. ISSN: 0004-6361. DOI: `10.1051/0004-6361/201731195`. URL: `http://adsabs.harvard.edu/abs/2017A%26A...608A...2I` (visited on 04/16/2021).

[29] K-*d Tree*. In: *Wikipedia*. Mar. 6, 2021. URL: `https://en.wikipedia.org/w/index.php?title=K-d_tree&oldid=1010710140` (visited on 03/12/2021).

[30] Thomas Kluyver et al. “Jupyter Notebooks – a Publishing Format for Reproducible Computational Workflows”. In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas* (2016), pp. 87–90. DOI: `10.3233/978-1-61499-649-1-87`. URL: `https://ebooks.iospress.nl/doi/10.3233/978-1-61499-649-1-87` (visited on 04/09/2021).

[31] John Kormendy and S. Djorgovski. “Surface Photometry and the Structure of Elliptical Galaxies”. In: *Annual Review of Astronomy and Astrophysics* 27 (1989), pp. 235–277. ISSN: 0066-4146. DOI: `10.1146/annurev.aa.27.090189.001315`. URL: `http://adsabs.harvard.edu/abs/1989ARA%26A..27..235K` (visited on 03/03/2021).

[32] Pierre Léna, Francois Lebrun, and Francois Mignard. *Observational Astrophysics*. 2nd ed. Astronomy and Astrophysics Library. Berlin Heidelberg: Springer-Verlag, 1998. ISBN: 978-3-642-08336-5. DOI: `10.1007/978-3-662-03685-3`. URL: `https://www.springer.com/gp/book/9783642083365` (visited on 05/09/2021).

[33] J Lewis et al. "DER SNR: A Simple & General Spectroscopic Signal-to-Noise Measurement Algorithm". In: (), p. 4.

[34] David Mary et al. "ORIGIN: Blind Detection of Faint Emission Line Galaxies in MUSE Datacubes". In: *Astronomy & Astrophysics* 635 (Mar. 1, 2020), A194. ISSN: 0004-6361, 1432-0746. DOI: `10.1051/0004-6361/201937001`. URL: `https://www.aanda.org/articles/aa/abs/2020/03/aa37001-19/aa37001-19.html` (visited on 03/10/2021).

[35] M. Masias et al. "A Review of Source Detection Approaches in Astronomical Images". In: *Monthly Notices of the Royal Astronomical Society* 422.2 (May 11, 2012), pp. 1674–1689. ISSN: 0035-8711. DOI: `10.1111/j.1365-2966.2012.20742.x`. URL: `https://doi.org/10.1111/j.1365-2966.2012.20742.x` (visited on 02/05/2021).

[36] *Modeling the Background — SExtractor 2.24.2 Documentation.* URL: `https://sextractor.readthedocs.io/en/latest/Background.html#equation-sexbackmode` (visited on 03/12/2021).

[37] *MUSELET — MPDAF 3.6.Dev15+gf936957 Documentation.* URL: `https://mpdaf.readthedocs.io/en/latest/muselet.html` (visited on 03/12/2021).

[38] Donald E. Osterbrock et al. "Night-Sky High-Resolution Spectral Atlas of OH and O2 Emission Lines for Echelle Spectrograph Wavelength Calibration". In: *Publications of the Astronomical Society of the Pacific* 108 (Mar. 1, 1996), p. 277. ISSN: 0004-6280. DOI: `10.1086/133722`. URL: `http://adsabs.harvard.edu/abs/1996PASP..108..277O` (visited on 04/12/2021).

[39] Fabian Pedregosa et al. "Scikit-Learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830. URL: `http://jmlr.org/papers/v12/pedregosa11a.html` (visited on 04/30/2021).

[40] W. D. Pence et al. "Definition of the Flexible Image Transport System (FITS), Version 3.0". In: *Astronomy & Astrophysics* 524 (Dec. 1, 2010), A42. ISSN: 0004-6361, 1432-0746. DOI: `10.1051/0004-6361/201015362`. URL: `https://www.aanda.org/articles/aa/abs/2010/16/aa15362-10/aa15362-10.html` (visited on 03/17/2021).

[41] Jeff Reback et al. *Pandas-Dev/Pandas: Pandas 1.2.3.* Zenodo, Mar. 2, 2021. DOI: `10.5281/zenodo.4572994`. URL: `https://zenodo.org/record/4572994` (visited on 03/12/2021).

[42] T. Sakamoto et al. "A Fast Asteroid Detection Algorithm". In: 434 (Dec. 1, 2010), p. 394. URL: `http://adsabs.harvard.edu/abs/2010ASPC..434..394S` (visited on 03/12/2021).

[43] Katharine J. Schlesinger et al. "The Metallicity Distribution Functions of SEGUE G and K Dwarfs: Constraints for Disk Chemical Evolution and Formation". In: *The Astrophysical Journal* 761 (Dec. 1, 2012), p. 160. ISSN: 0004-637X. DOI: `10.1088/0004-637X/761/2/160`. URL: `http://adsabs.harvard.edu/abs/2012ApJ...761..160S` (visited on 05/07/2021).

[44] Kurt T. Soto et al. "ZAP: Zurich Atmosphere Purge". In: *Astrophysics Source Code Library* (Feb. 1, 2016), ascl:1602.003. URL: `http://adsabs.harvard.edu/abs/2016ascl.soft02003S` (visited on 04/28/2021).

[45] *Source Extractor Wrapper for Python — Sewpy 1.0dev Documentation.* URL: `https://sewpy.readthedocs.io/en/latest/` (visited on 03/12/2021).

[46] Peter B. Stetson. “DAOPHOT - A Computer Program for Crowded-Field Stellar Photometry”. In: *Publications of the Astronomical Society of the Pacific* 99 (Mar. 1, 1987), pp. 191–222. ISSN: 0004-6280. DOI: `10.1086/131977`. URL: `http://adsabs.harvard.edu/abs/1987PASP...99..191S` (visited on 03/18/2021).

[47] Max Tegmark et al. “Cosmological Parameters from SDSS and WMAP”. In: *Physical Review D* 69.10 (May 5, 2004), p. 103501. ISSN: 1550-7998, 1550-2368. DOI: `10.1103/PhysRevD.69.103501`. URL: `https://link.aps.org/doi/10.1103/PhysRevD.69.103501` (visited on 05/07/2021).

[48] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17.3 (3 Mar. 2020), pp. 261–272. ISSN: 1548-7105. DOI: `10.1038/s41592-019-0686-2`. URL: `https://www.nature.com/articles/s41592-019-0686-2` (visited on 03/12/2021).

[49] Peter M. Weilbacher et al. “The Data Processing Pipeline for the MUSE Instrument”. In: *Astronomy & Astrophysics* 641 (Sept. 1, 2020), A28. ISSN: 0004-6361, 1432-0746. DOI: `10.1051/0004-6361/202037855`. URL: `https://www.aanda.org/articles/aa/abs/2020/09/aa37855-20/aa37855-20.html` (visited on 03/17/2021).

[50] L. Wisotzki et al. “Nearly All the Sky Is Covered by Lyman-α Emission around High-Redshift Galaxies”. In: *Nature* 562.7726 (7726 Oct. 2018), pp. 229–232. ISSN: 1476-4687. DOI: `10.1038/s41586-018-0564-6`. URL: `https://www.nature.com/articles/s41586-018-0564-6` (visited on 03/17/2021).

[51] D. S. Womble, W. L. W. Sargent, and R. S. Lyons. “Heavy Elements in the Lyman Alpha Forest: Abundances and Clustering at Z=3”. In: *Cold Gas at High Redshift* (1996), pp. 249–253. DOI: `10.1007/978-94-009-1726-2_27`. URL: `https://link.springer.com/chapter/10.1007/978-94-009-1726-2_27` (visited on 04/12/2021).

[52] Brian Yanny et al. “SEGUE: A SPECTROSCOPIC SURVEY OF 240,000 STARS WITH g = 14-20”. In: *The Astronomical Journal* 137.5 (May 1, 2009), pp. 4377–4399. ISSN: 0004-6256, 1538-3881. DOI: `10.1088/0004-6256/137/5/4377`. URL: `https://iopscience.iop.org/article/10.1088/0004-6256/137/5/4377` (visited on 05/07/2021).

[53] *Zooniverse*. URL: `https://www.zooniverse.org/projects/zookeeper/galaxy-zoo/about/research` (visited on 03/04/2021).

Appendix A

Plots and spectral matches from the Hubble Ultra Deep Field

In this appendix, we present various plots from the Hubble Ultra Deep Field, and their correlated Marz templates.

A.1 QOP 4

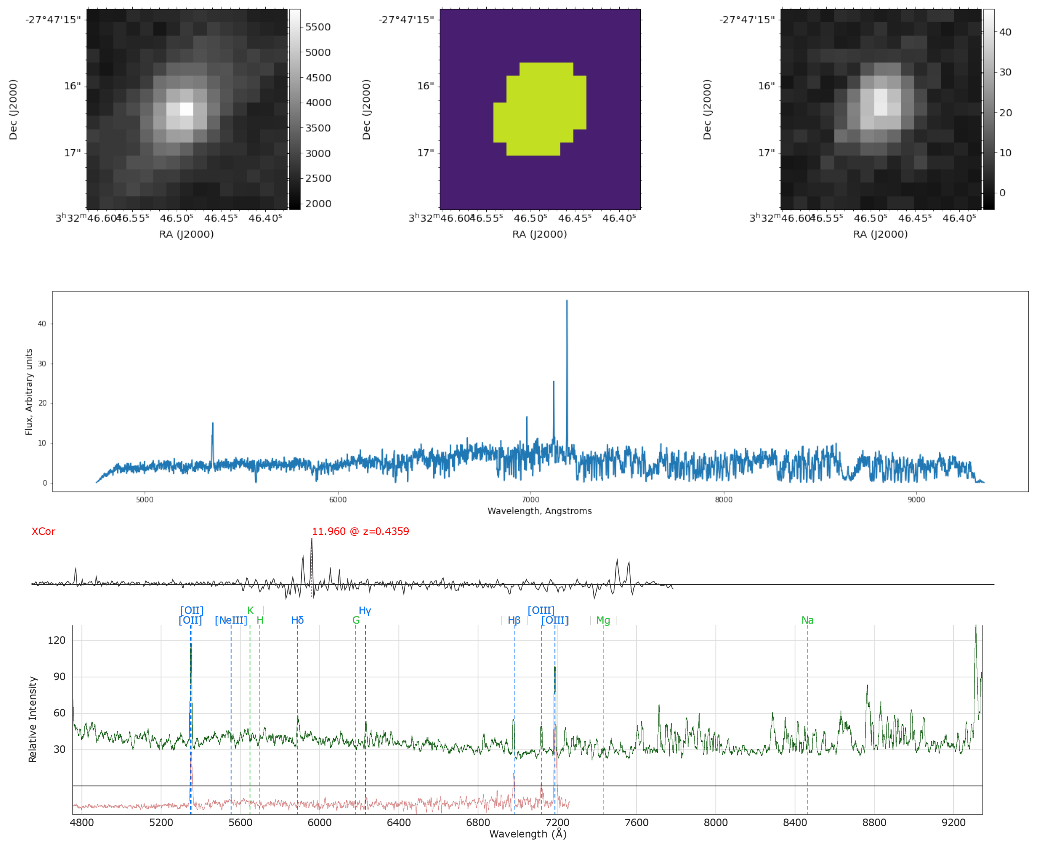

Figure A.1: Object 127 from HUDF and its Marz-template match, with QOP 4

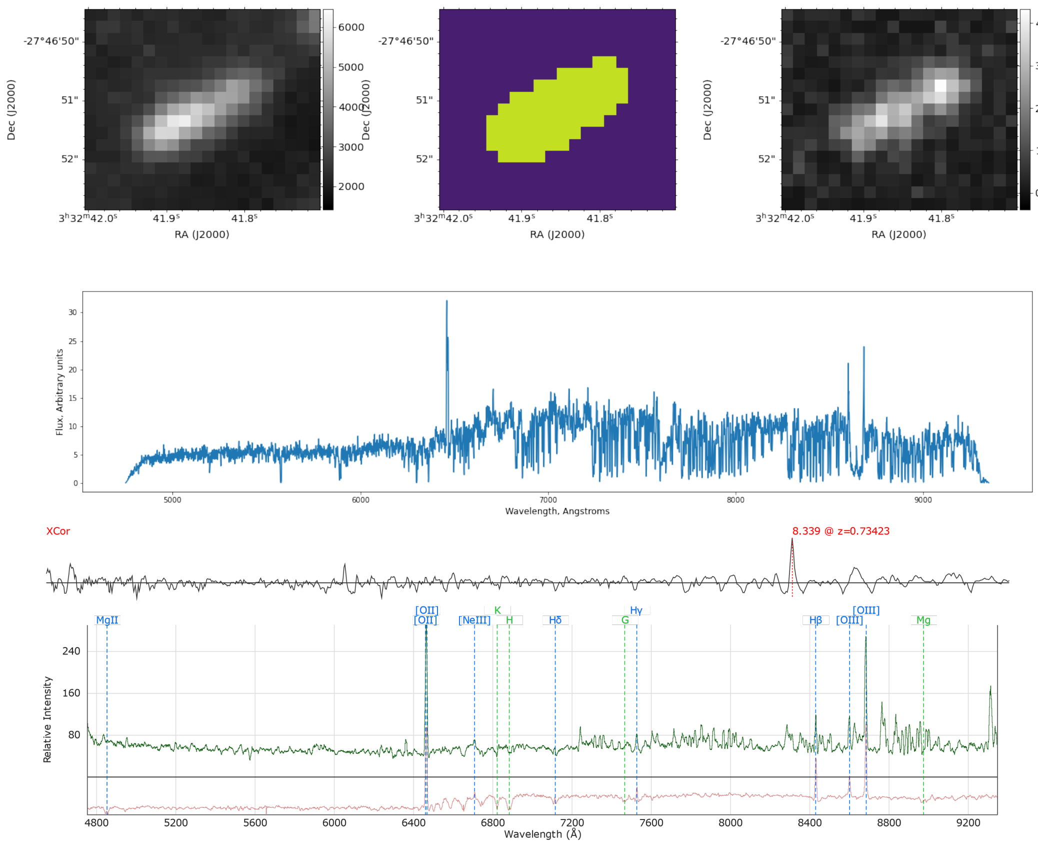

Figure A.2: Object 185 from HUDF and its Marz-template match, with QOP 4

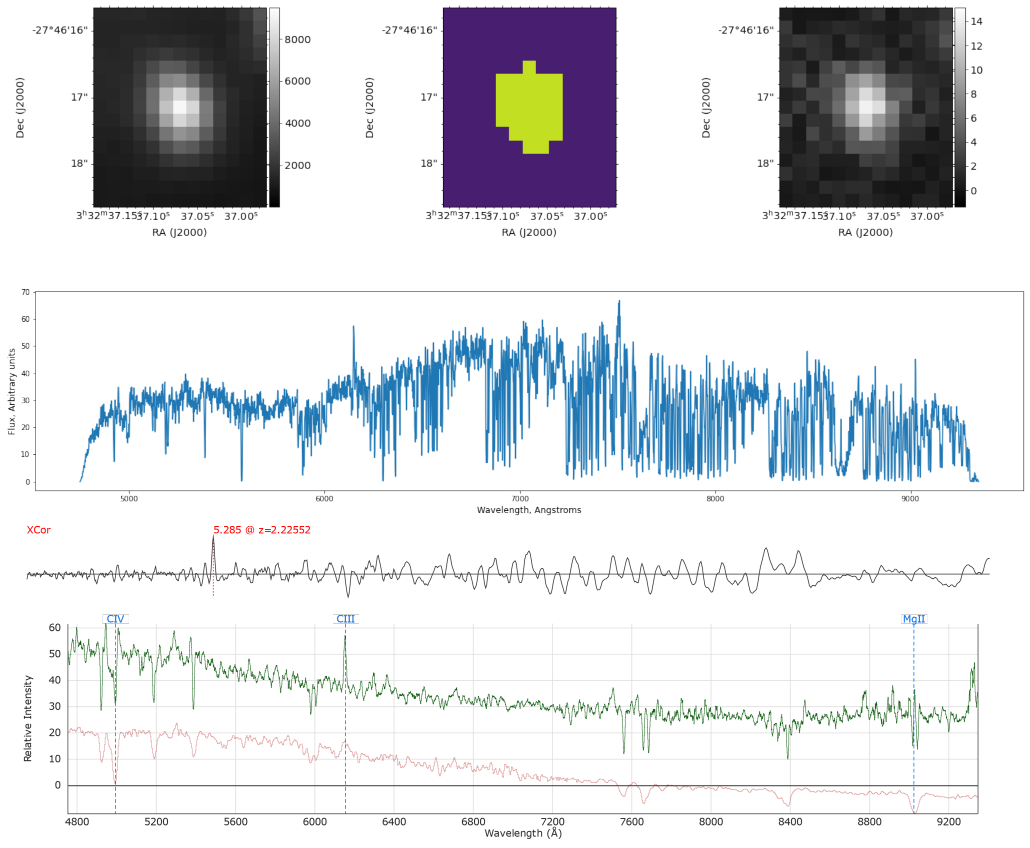

Figure A.3: Object 229 from HUDF and its Marz-template match, with QOP 4

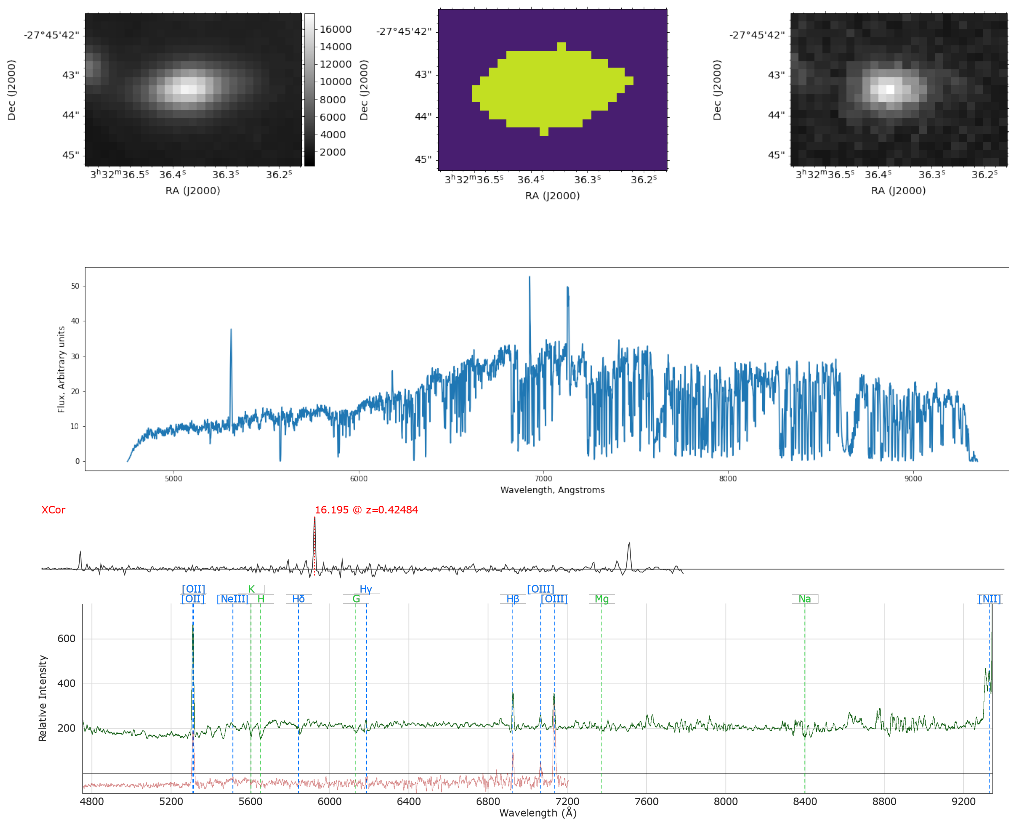

Figure A.4: Object 262 from HUDF and its Marz-template match, with QOP 4

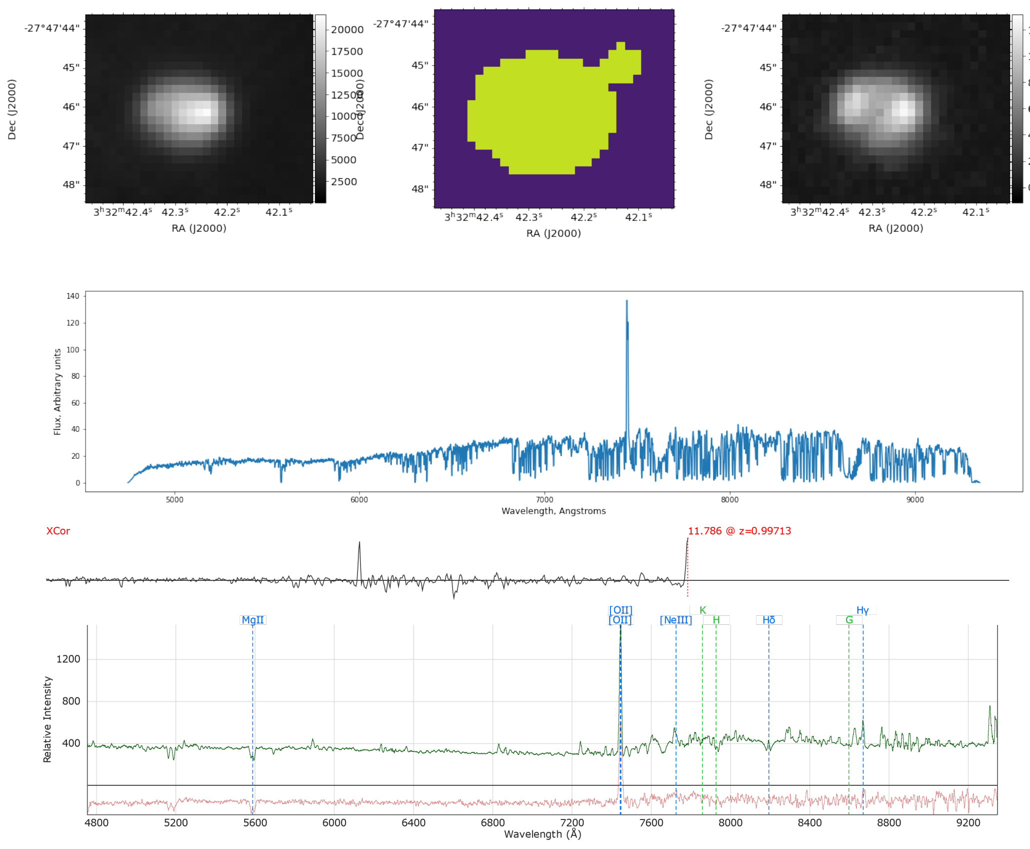

Figure A.5: Object 364 from HUDF and its Marz-template match, with QOP 4

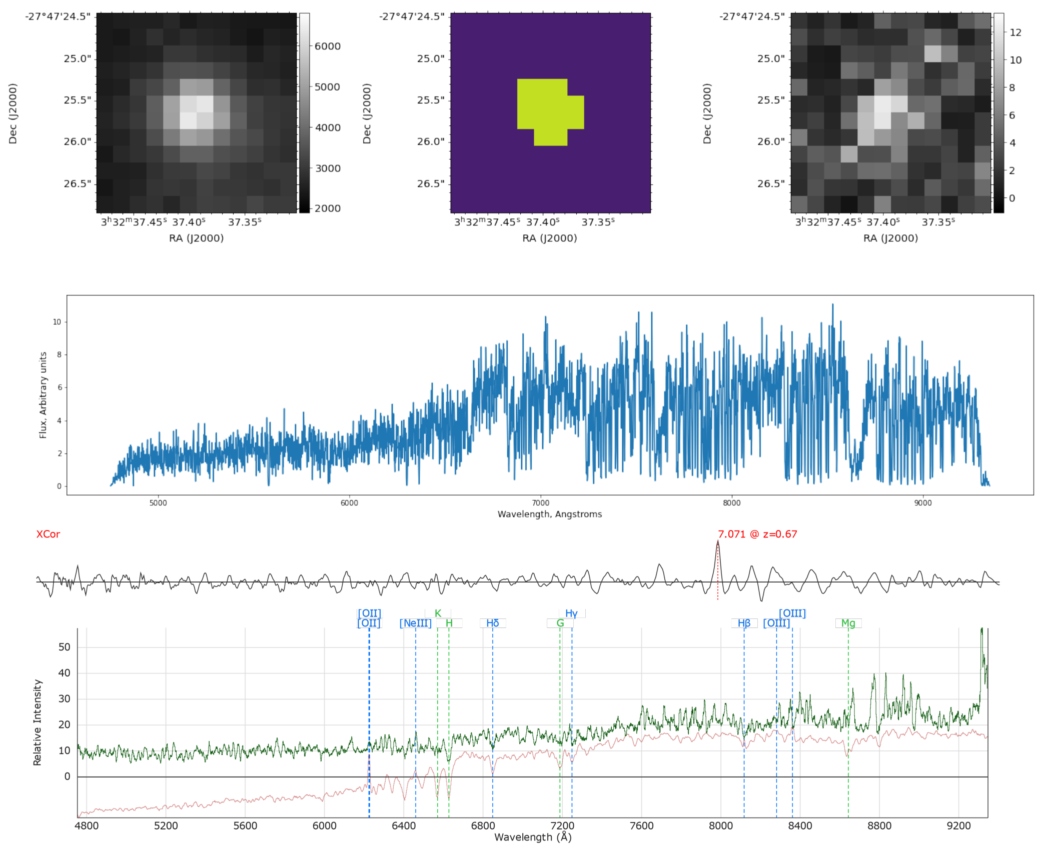

Figure A.6: Object 401 from HUDF and its Marz-template match, with QOP 4

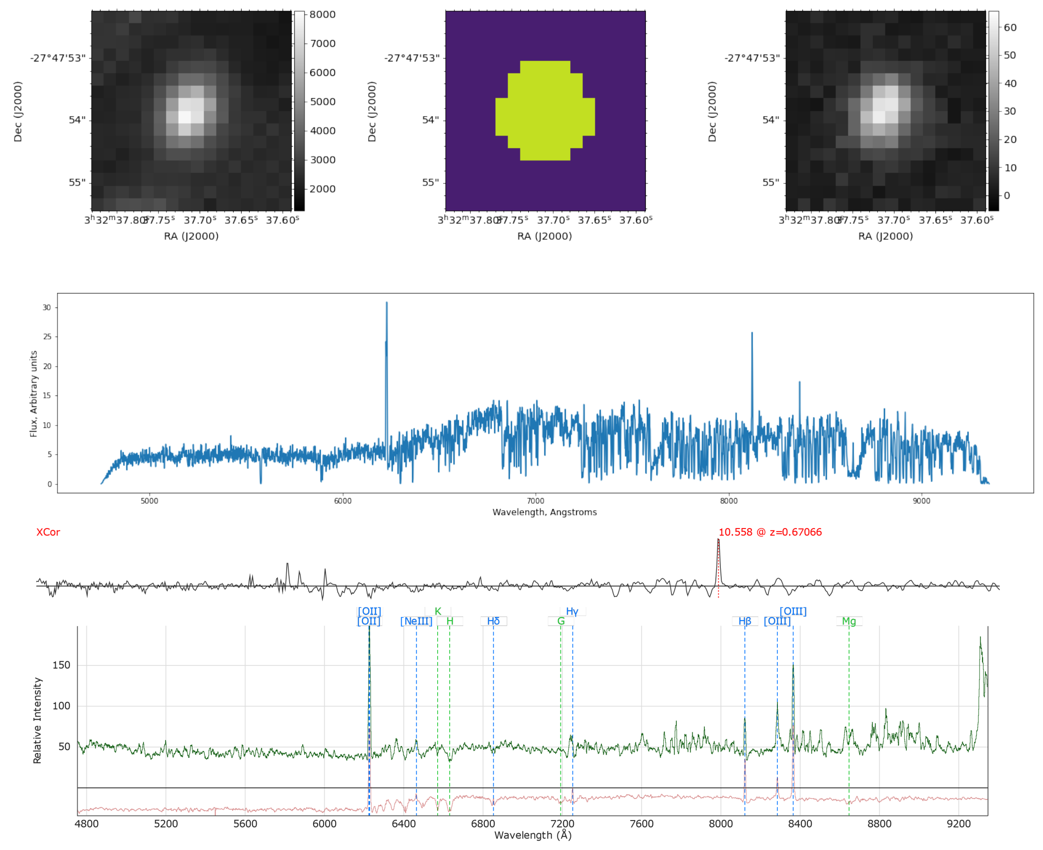

Figure A.7: Object 57 from HUDF and its Marz-template match, with QOP 4

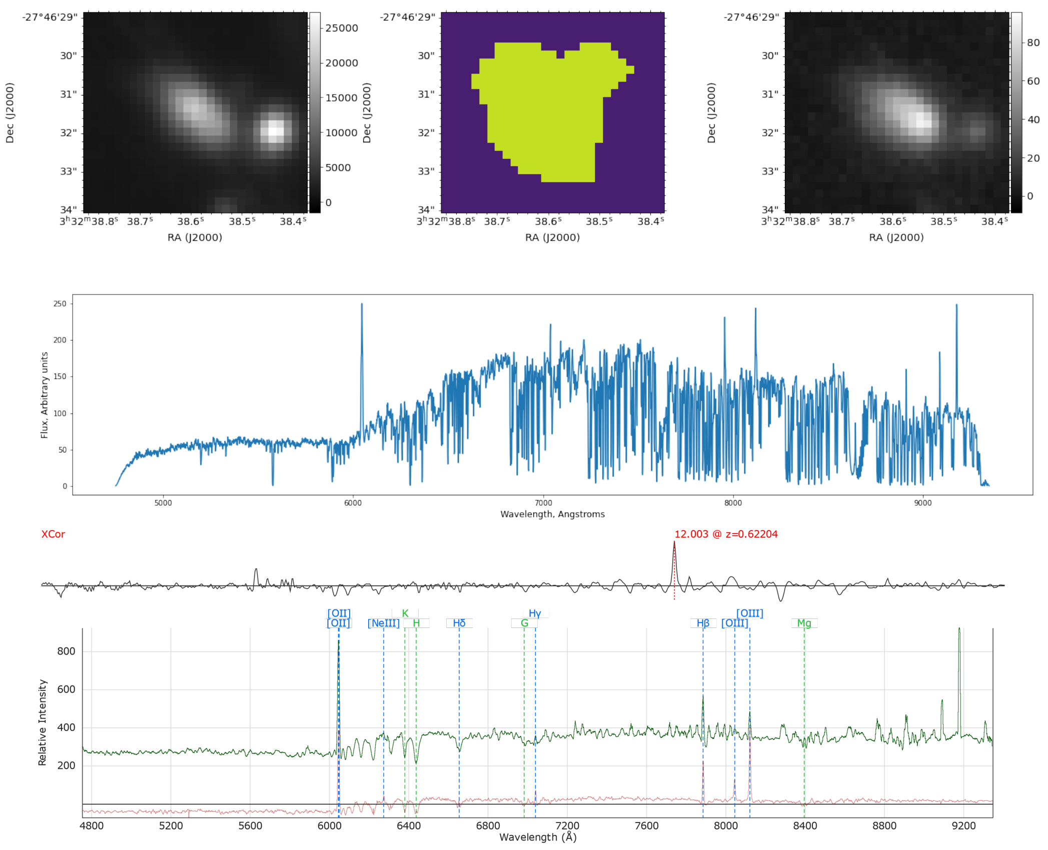

Figure A.8: Object 611 from HUDF and its Marz-template match, with QOP 4

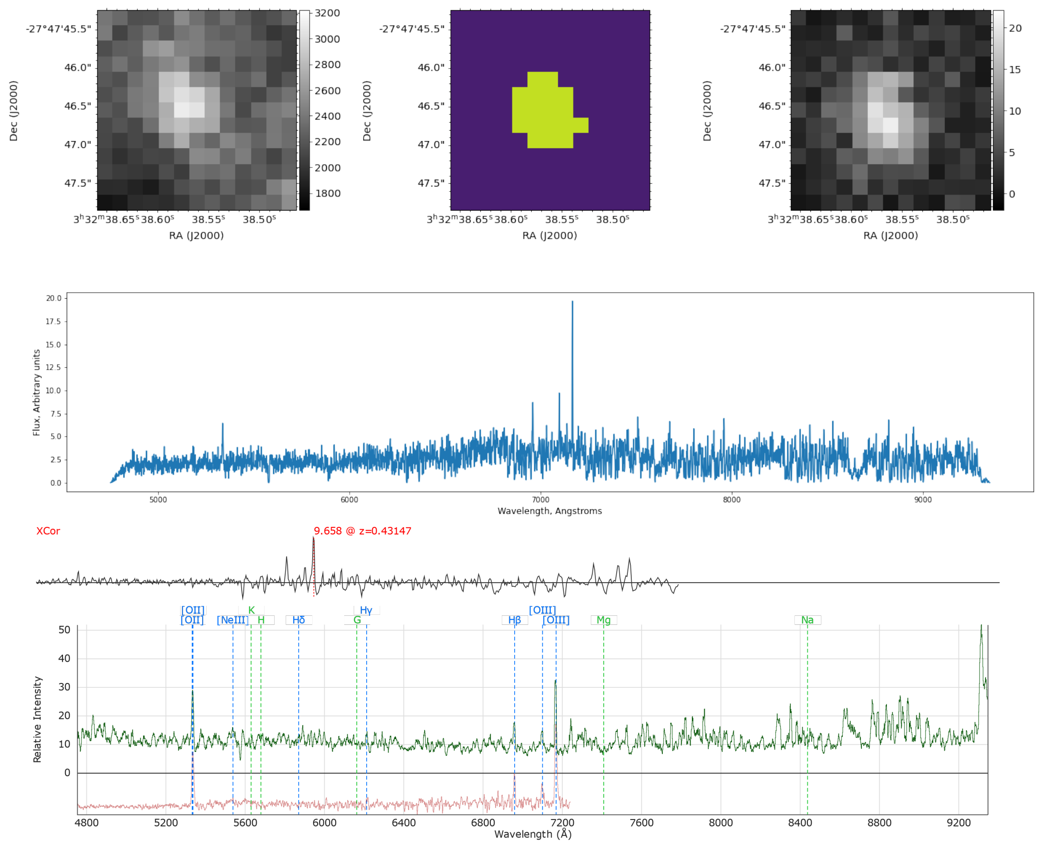

Figure A.9: Object 66 from HUDF and its Marz-template match, with QOP 4

A.2 QOP 3

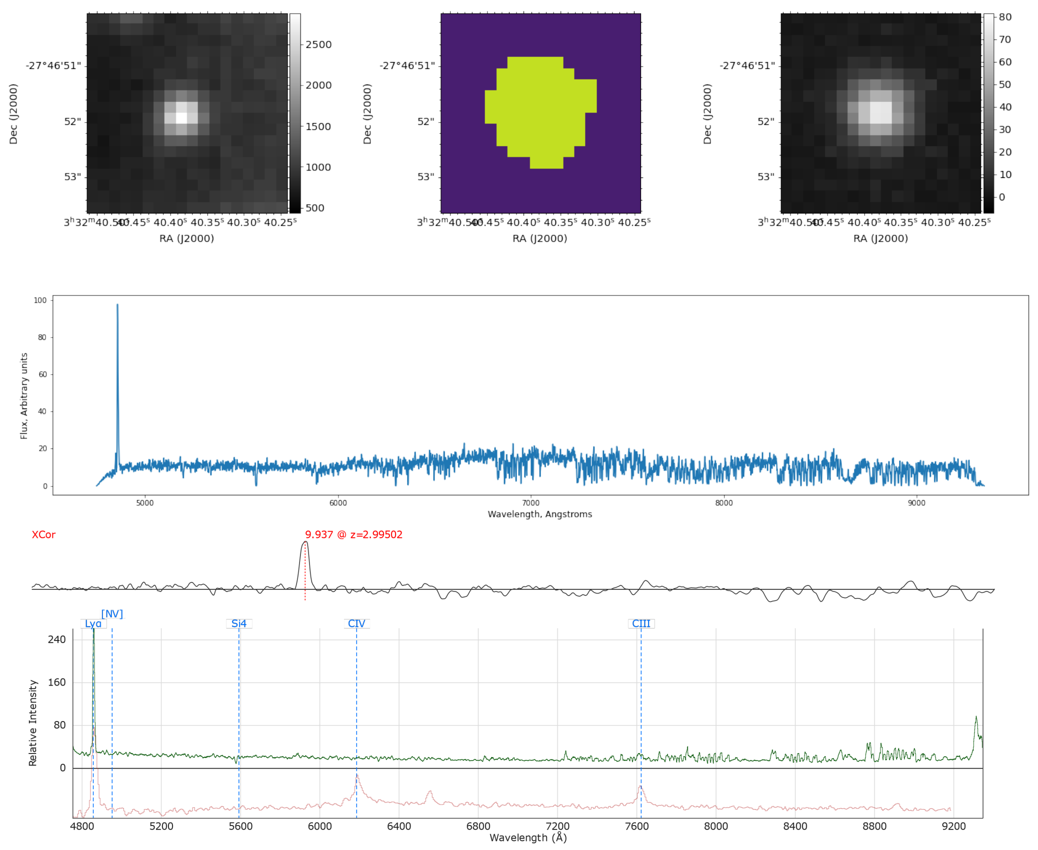

Figure A.10: Object 181 from HUDF and its Marz-template match, with QOP 3

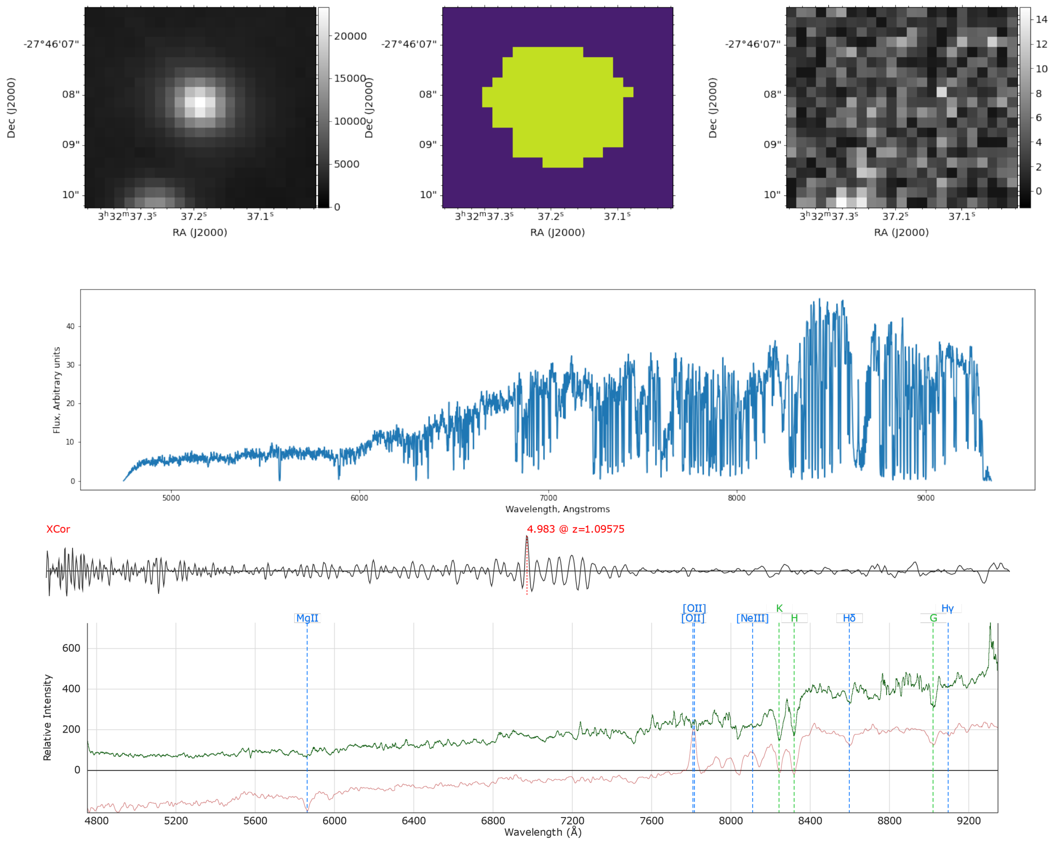

Figure A.11: Object 293 from HUDF and its Marz-template match, with QOP 3

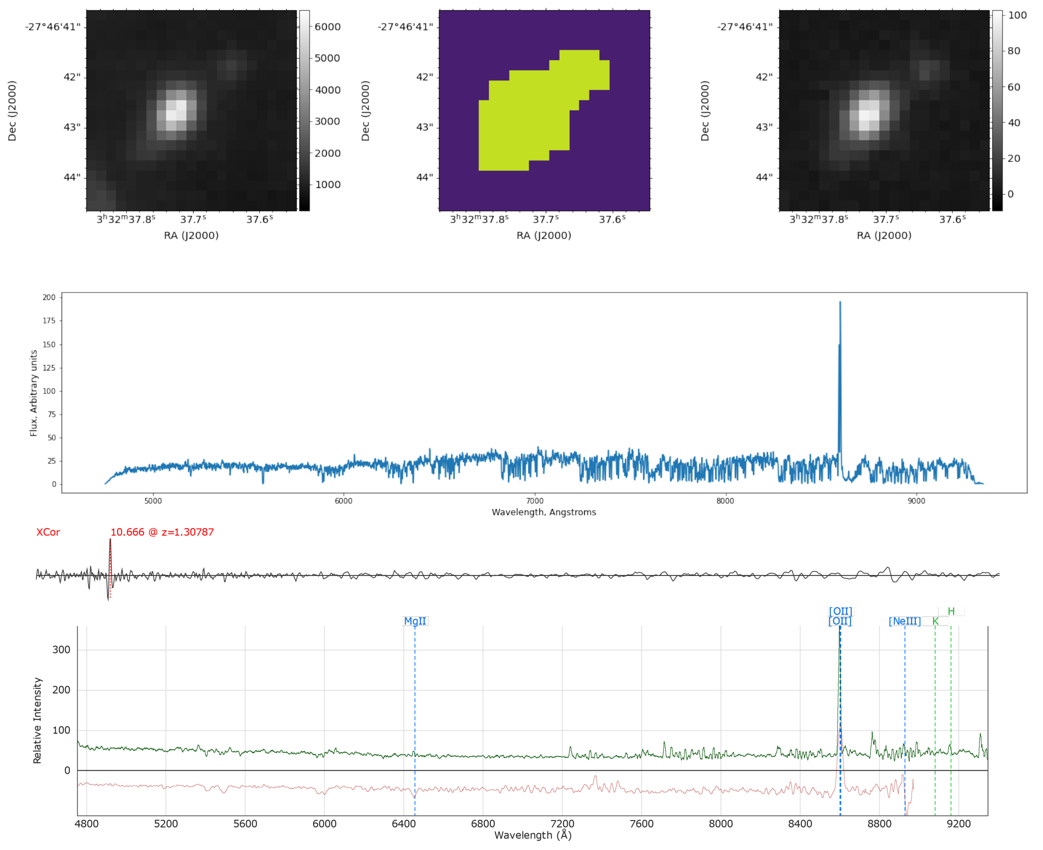

Figure A.12: Object 496 from HUDF and its Marz-template match, with QOP 3

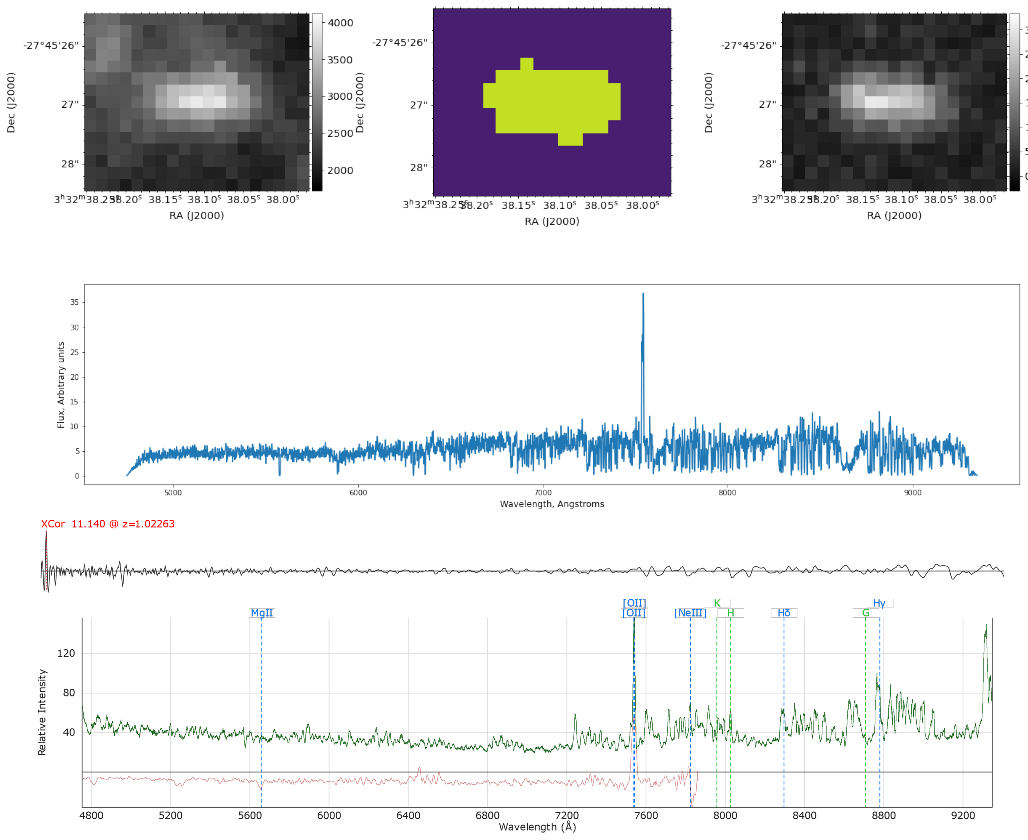

Figure A.13: Object 588 from HUDF and its Marz-template match, with QOP 3

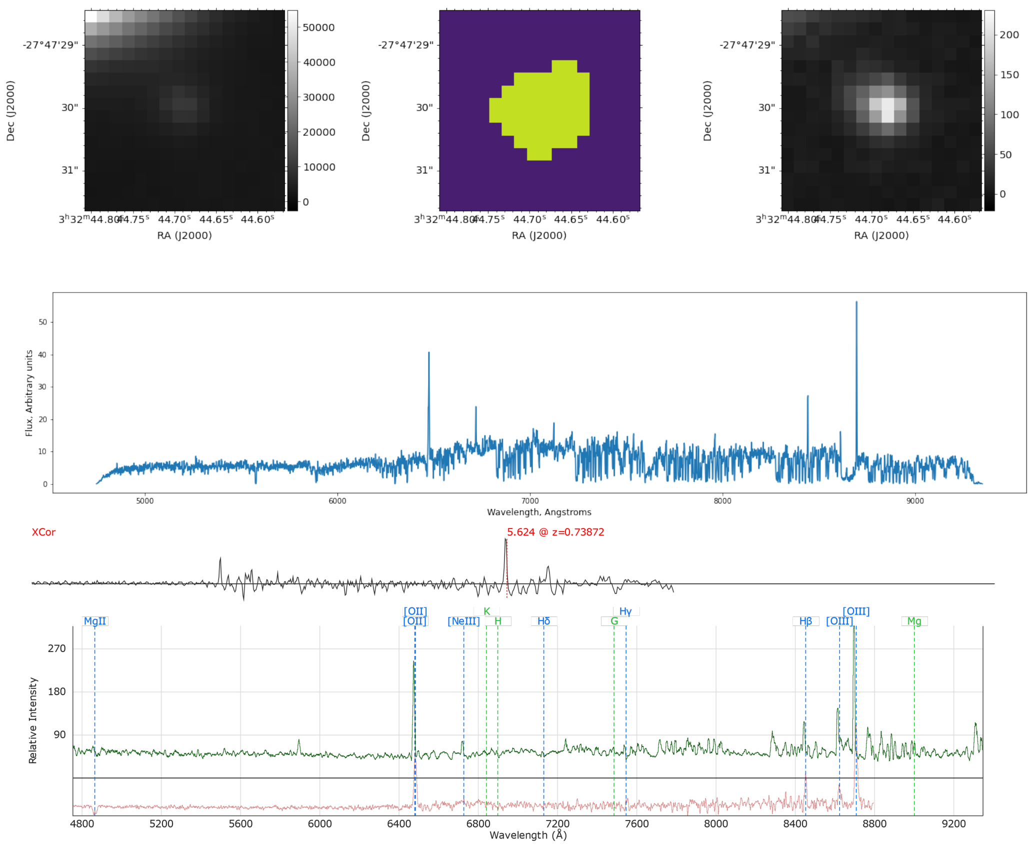

Figure A.14: Object 609 from HUDF and its Marz-template match, with QOP 3

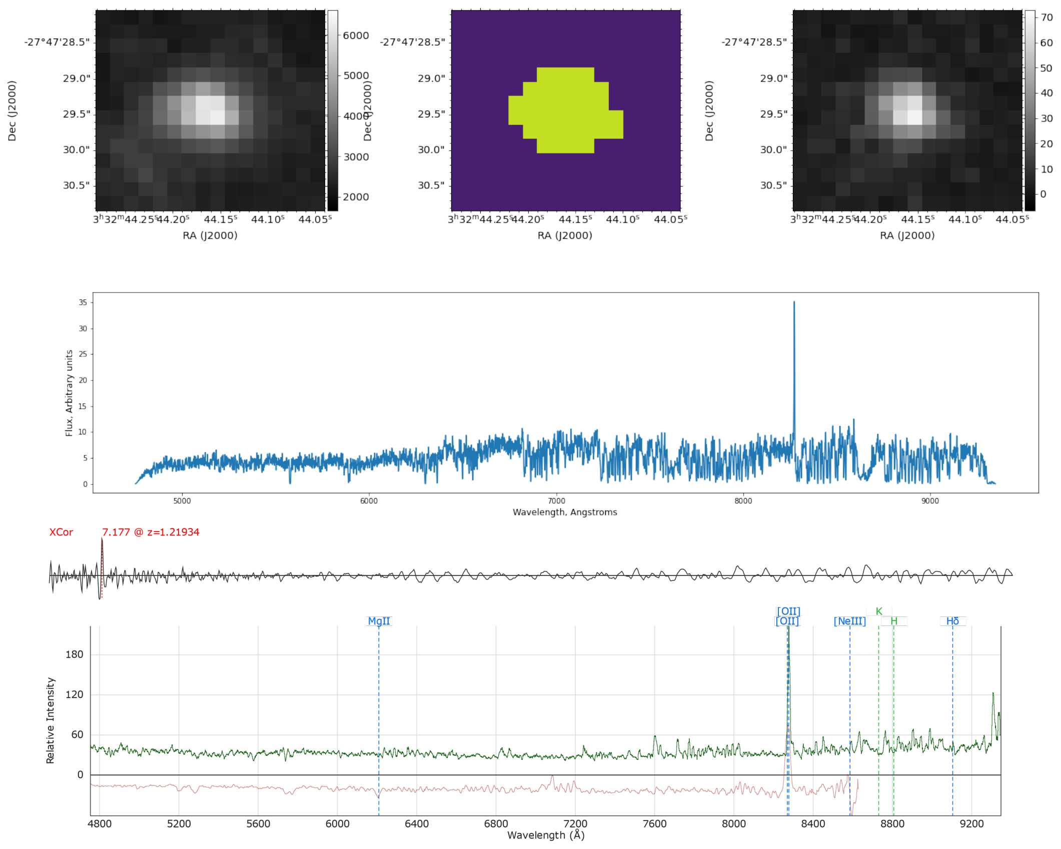

Figure A.15: Object 97 from HUDF and its Marz-template match, with QOP 3

A.3 QOP 2

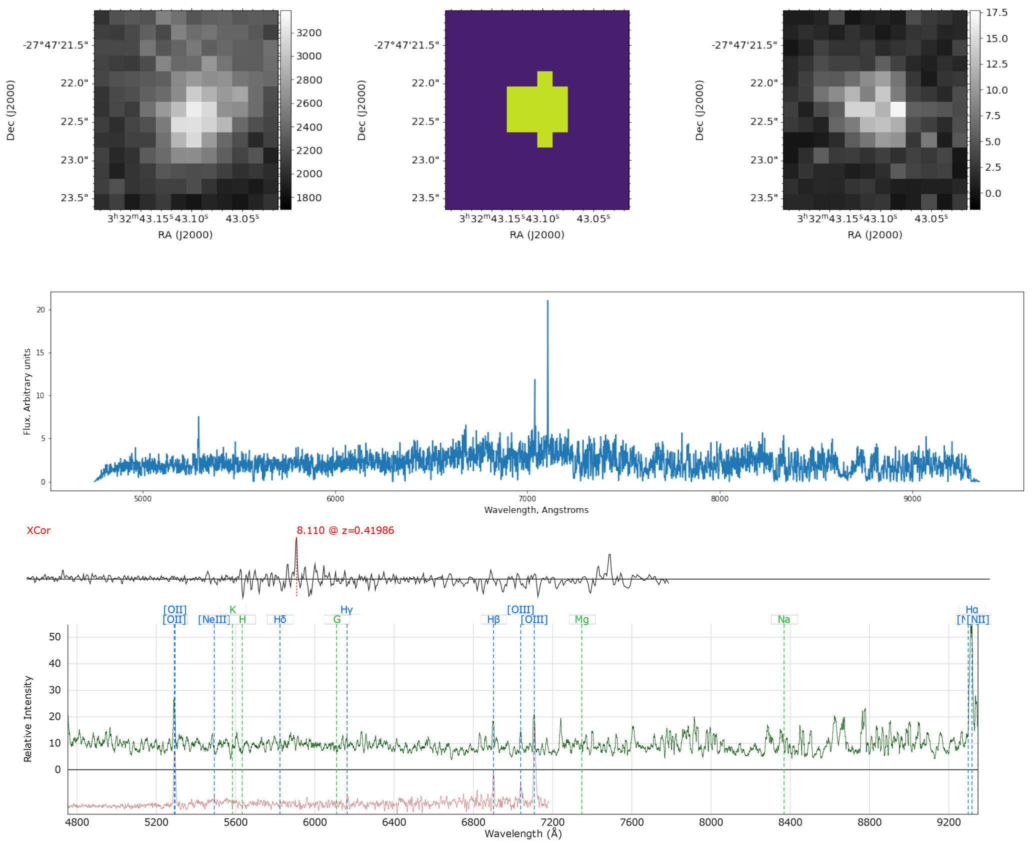

Figure A.16: Object 116 from HUDF and its Marz-template match, with QOP 2

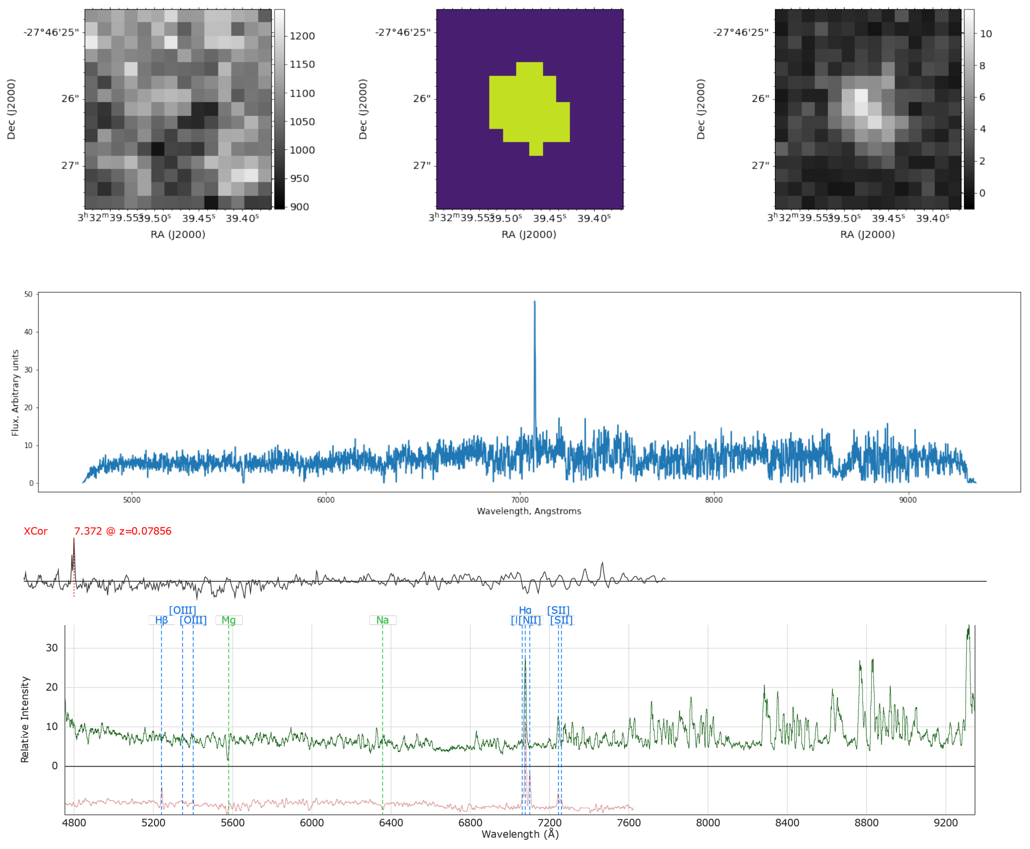

Figure A.17: Object 218 from HUDF and its Marz-template match, with QOP 2

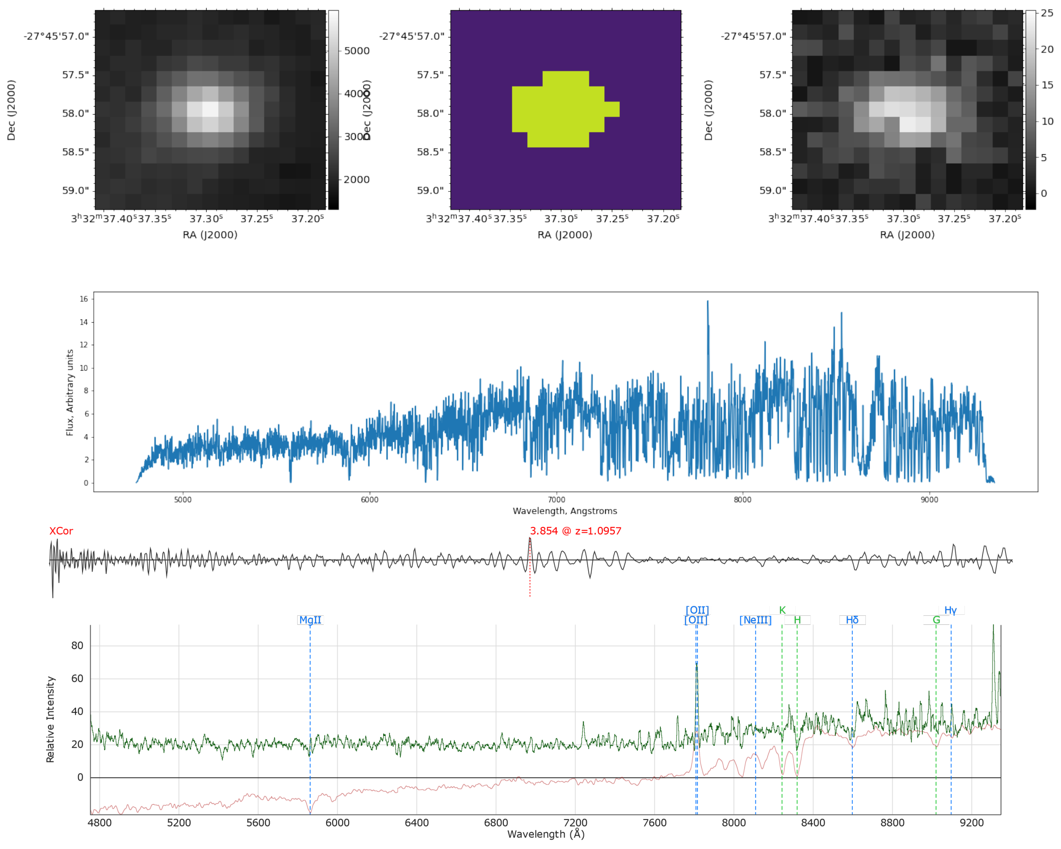

Figure A.18: Object 253 from HUDF and its Marz-template match, with QOP 2

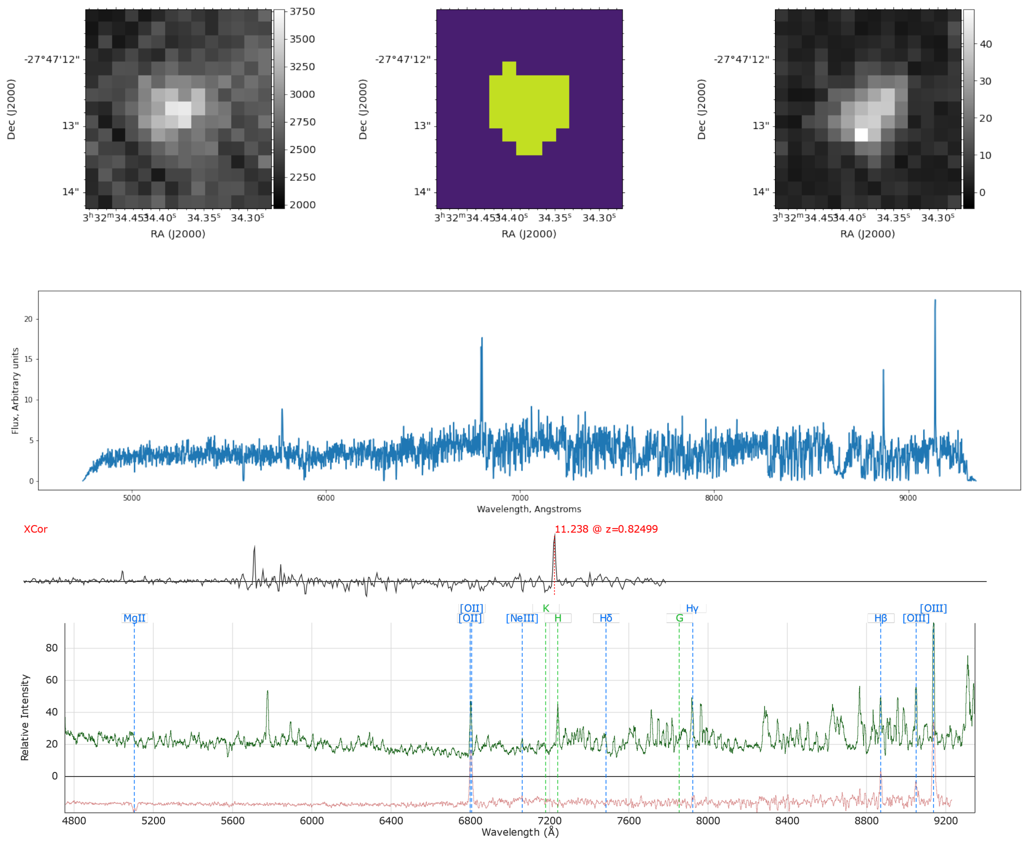

Figure A.19: Object 430 from HUDF and its Marz-template match, with QOP 2

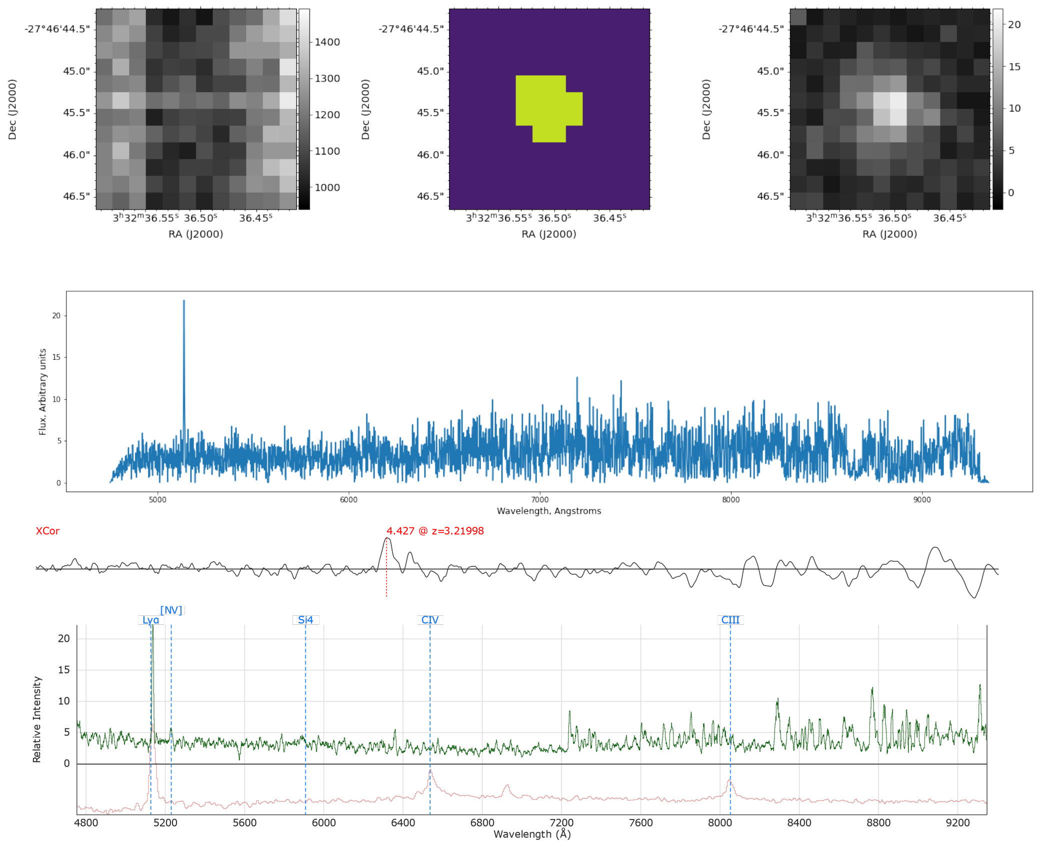

Figure A.20: Object 492 from HUDF and its Marz-template match, with QOP 2

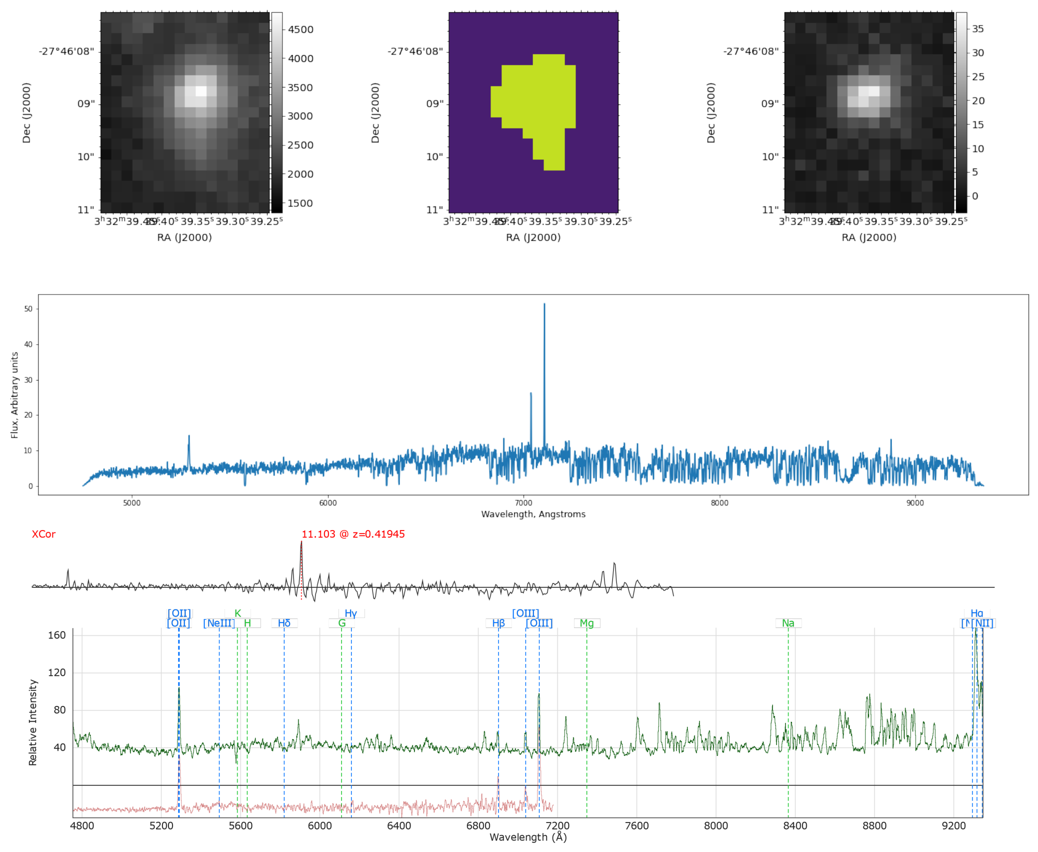

Figure A.21: Object 554 from HUDF and its Marz-template match, with QOP 2

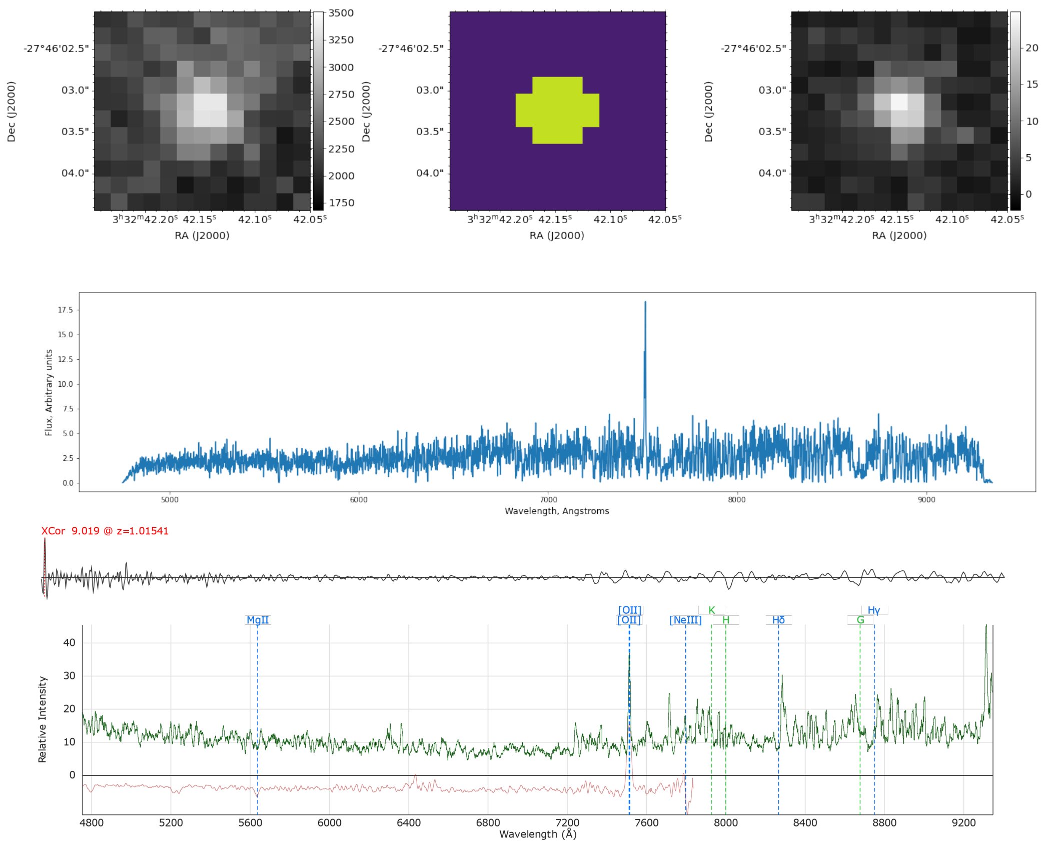

Figure A.22: Object 565 from HUDF and its Marz-template match, with QOP 2

A.4 QOP 1

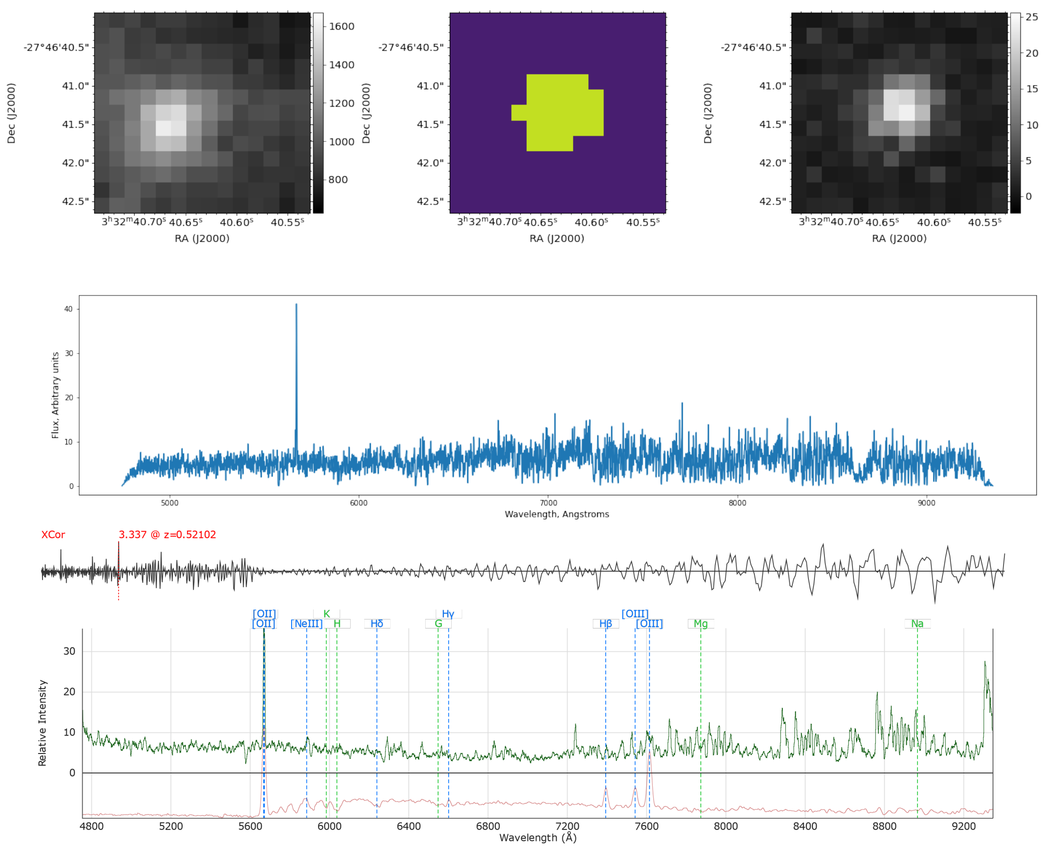

Figure A.23: Object 200 from HUDF and its Marz-template match, with QOP 1

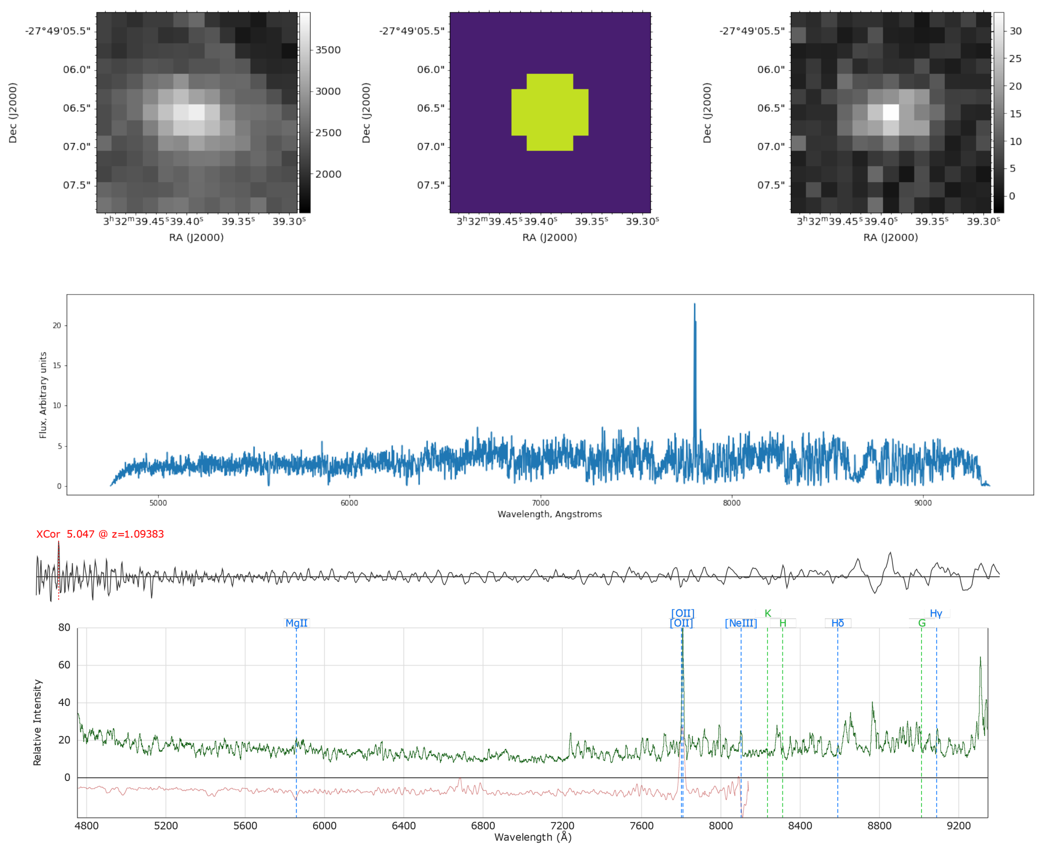

Figure A.24: Object 306 from HUDF and its Marz-template match, with QOP 1

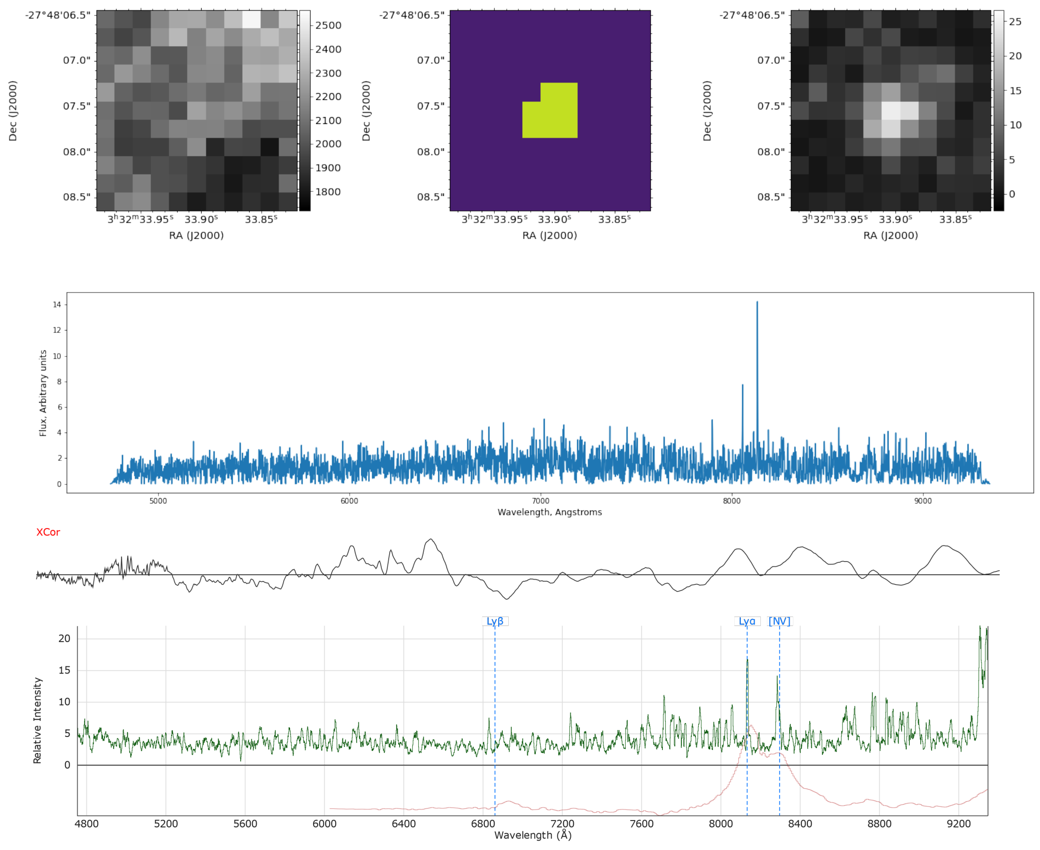

Figure A.25: Object 337 from HUDF and its Marz-template match, with QOP 1

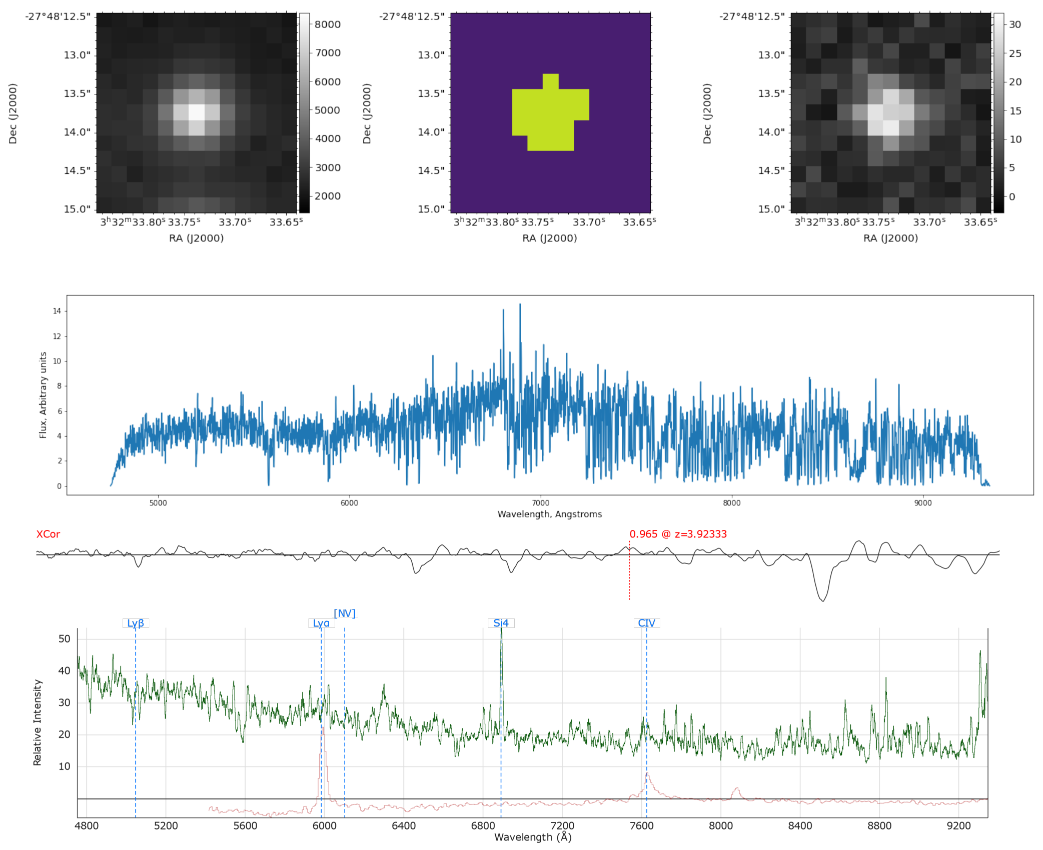

Figure A.26: Object 35 from HUDF and its Marz-template match, with QOP 1

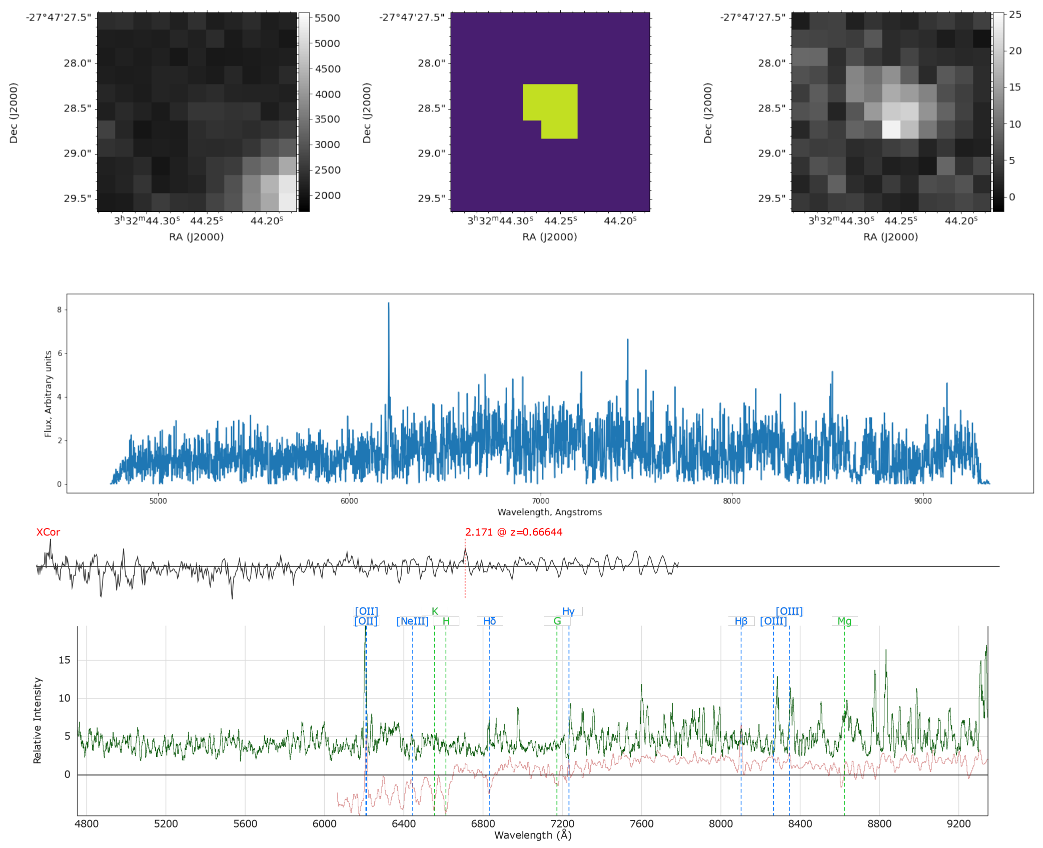

Figure A.27: Object 395 from HUDF and its Marz-template match, with QOP 1

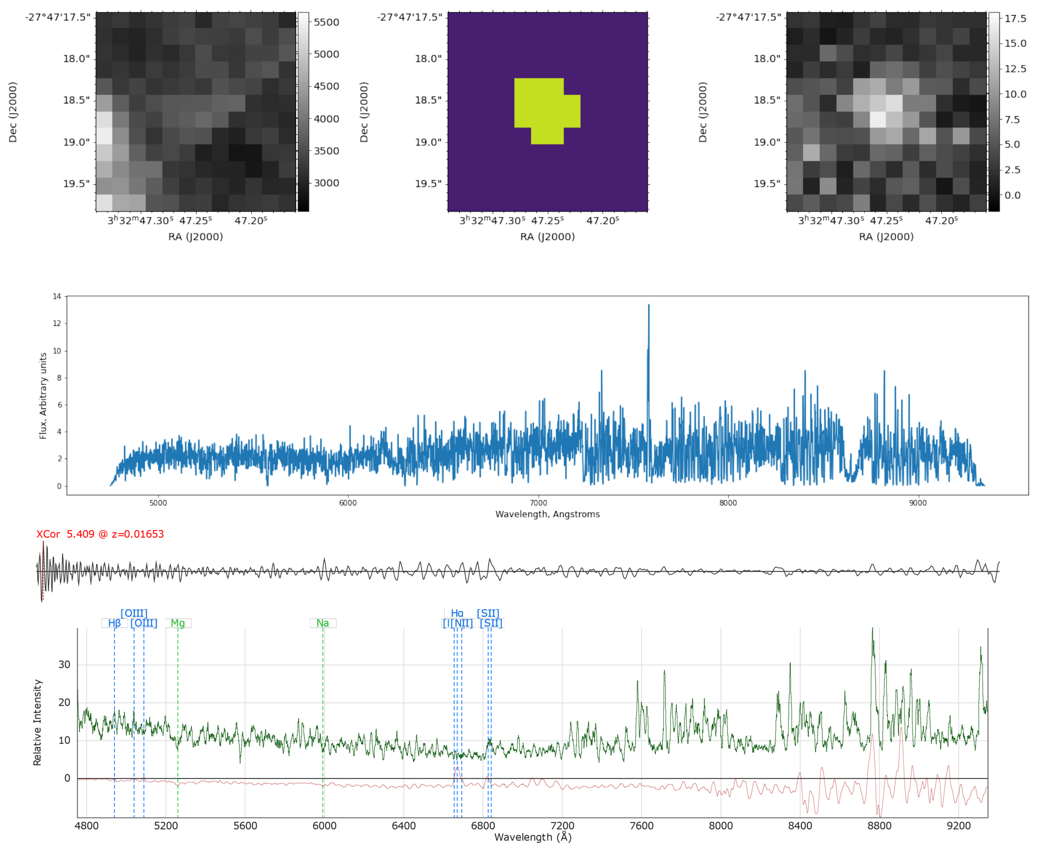

Figure A.28: Object 419 from HUDF and its Marz-template match, with QOP 1

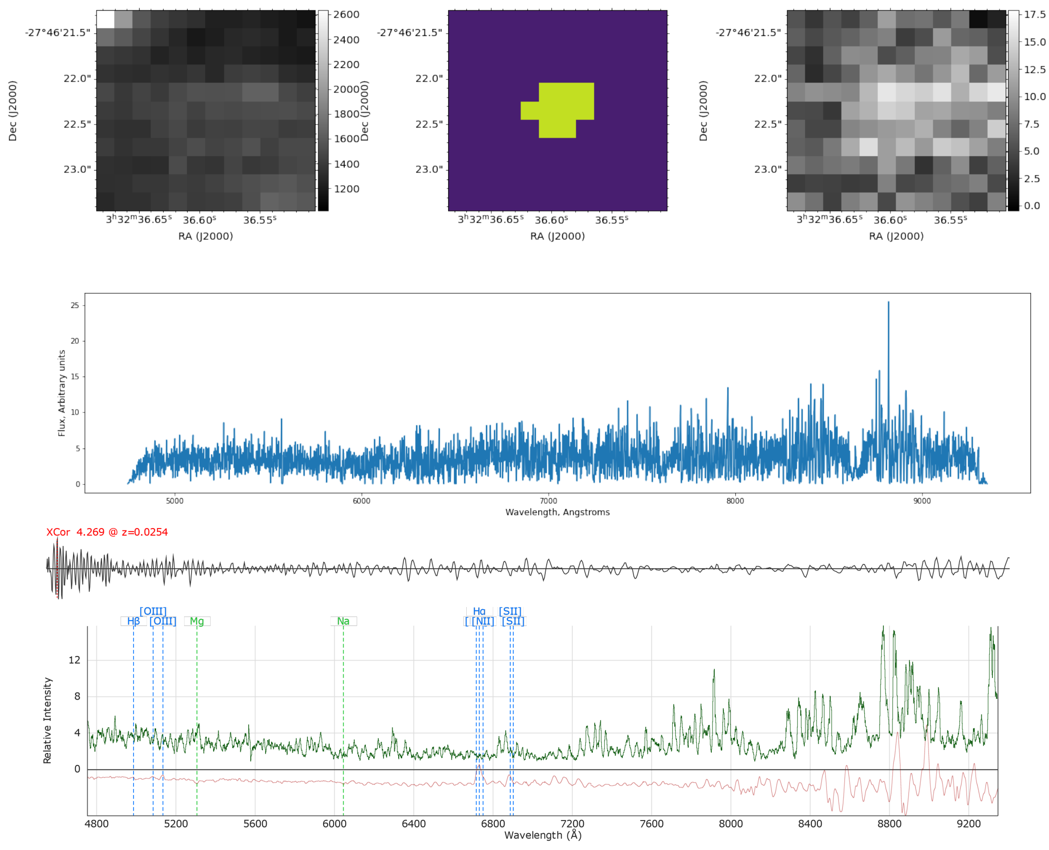

Figure A.29: Object 536 from HUDF and its Marz-template match, with QOP 1

Appendix B

Python object plotter code

The following code is used to plot images from objects, outputting white-light, source segmentation, narrow-band image of the brightest emission line, and spectrum of the object (tapered with a cosine windows). It is the code generating images along the thesis, and the ones in Appendix A.

```python
from astropy.io import fits
from astropy.nddata import CCDData
import matplotlib.pyplot as plt
from aplpy import FITSFigure
from scipy import signal
from scipy.ndimage import convolve1d
import numpy as np
import os
from astropy.utils.exceptions import AstropyWarning
import warnings
import aplpy
import ipywidgets as widgets
from IPython.display import clear_output
from bokeh.io import output_notebook
import pandas as pd

from astropy.modeling import models

from astropy import units as u

from specutils.spectra import Spectrum1D, SpectralRegion

from specutils.fitting import fit_generic_continuum

warnings.simplefilter('ignore', category=AstropyWarning)


def obj_plotter(cubename):
    print(os.path.basename(cubename))
    cube = CCDData.read(cubename)
    window = signal.windows.tukey(len(cube.data), alpha=0.05)
    cubetozero = (cube.data <= 0) | (cube.uncertainty.array <=
       0) | np.isnan(cube.data) | np.isnan(
        cube.uncertainty.array)
    cube_noneg = cube.data
    cube_var = cube.uncertainty.array
    cube_noneg[cubetozero] = 0
    cube_var[cubetozero] = 0
    cubetozero = cube.mask
    cube_forspec = cube_noneg.copy()
    cube_forspec[cubetozero] = 0
    cube_var[cubetozero] = 0
    spec = cube_forspec.sum(axis=(1, 2)) * window * cube.unit
    specvar = cube_var.sum(axis=(1, 2)) * window * cube.unit
    specaxis = cube.wcs.spectral.pixel_to_world(np.arange(0, len
       (spec)))
    spec = Spectrum1D(spec, specaxis)
    g1_fit = fit_generic_continuum(spec)
    y_continuum_fitted = g1_fit(specaxis)
    spec = (spec.flux / y_continuum_fitted / np.sqrt(specvar)).
       value
    spec[np.isnan(spec)] = 0
    bbcenter = np.where(spec == max(spec))[0][0]
    spec = spec * y_continuum_fitted
    nbsum = cube_noneg[bbcenter - 2:bbcenter + 3].sum(axis=0)
    wlsum = cube_noneg.sum(axis=0)
    wlHDU = fits.ImageHDU(data=wlsum, header=cube.wcs.celestial.
       to_header())
    segHDU = fits.ImageHDU(data=(~cube.mask[0]).astype(int),
       header=cube.wcs.celestial.to_header())
    nbHDU = fits.ImageHDU(data=nbsum, header=cube.wcs.celestial.
       to_header())
    fig = plt.figure(figsize=(20, 10))
    wl = FITSFigure(wlHDU, figure=fig, subplot=(2, 3, 1))
    wl.tick_labels.show()
    wl.tick_labels.set_font(size='x-large')
    wl.show_grayscale()
    wl.add_colorbar()
    wl.colorbar.set_font(size='x-large')
    wl.axis_labels.set_font(size='x-large')
    seg = FITSFigure(segHDU, figure=fig, subplot=(2, 3, 2))
    seg.tick_labels.show()
    seg.tick_labels.set_font(size='x-large')
    seg.show_colorscale()
    seg.axis_labels.set_font(size='x-large')
    nb = FITSFigure(nbHDU, figure=fig, subplot=(2, 3, 3))
    nb.tick_labels.show()
    nb.tick_labels.set_font(size='x-large')
    nb.show_grayscale()
    nb.add_colorbar()
    nb.colorbar.set_font(size='x-large')
    nb.axis_labels.set_font(size='x-large')
    # get axis bottom subplot
    axspec = fig.add_subplot(2, 3, (4, 6))
    axspec.plot(specaxis * 1e10, spec)
    axspec.set_ylabel("Flux, Arbitrary units", fontsize=12)
    axspec.set_xlabel("Wavelength, Angstroms", fontsize=12)
    fig.canvas.draw()
    plt.tight_layout(h_pad=7)
    fig.savefig('plots/' + os.path.basename(cubename)[:-5] + '.
       png')
    # plt.show(fig)
    plt.close(fig)
```

Appendix C

Additional segmentation maps analysis

We analyzed with AutoMUSE additional MUSE fields, targeting lensed quasars.The datacubes are not available publicly, and were obtained from Lise and Adriano. The left panel represents the segmentation map produced by AutoMUSE, while the right panel is the white-light image.

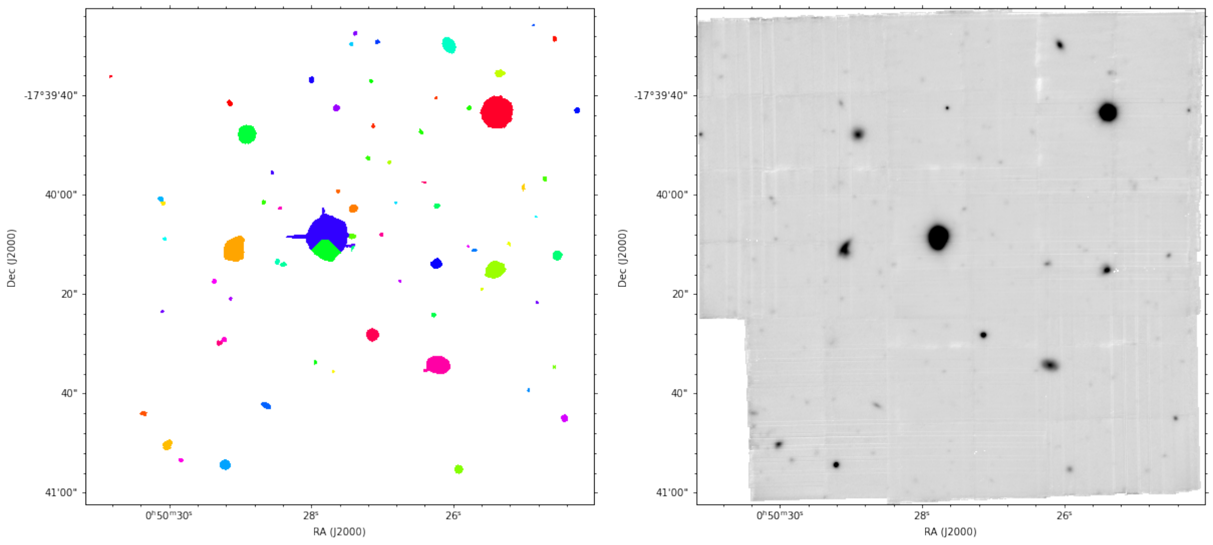

Figure C.1: Lensed quasar —HE0047— analyzed by AutoMUSE. There are 72 objects detected with detection threshold = 3. The target object “leaks”, and has small false detections around the center.

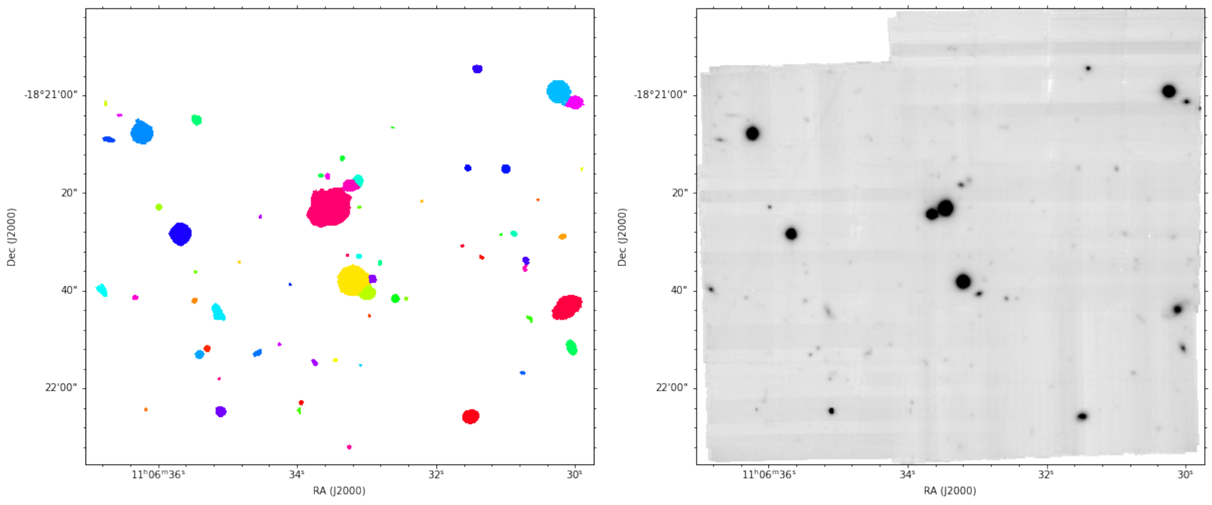

Figure C.2: Lensed quasar —HE1104— analyzed by AutoMUSE. There are 65 objects detected with detection threshold = 3. The target object, where we clearly see two sources on the white-light image, is not correctly deblended at this threshold.

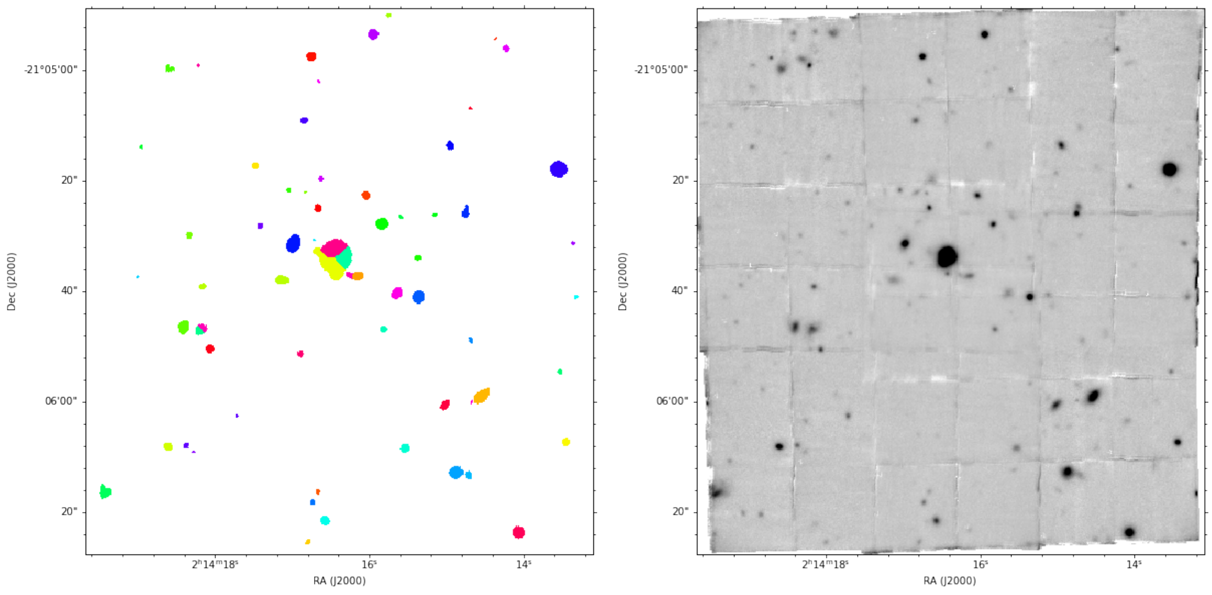

Figure C.3: Lensed quasar —J024— analyzed by AutoMUSE. There are 65 objects detected with detection threshold = 3. The target object seems to be correctly deblended.

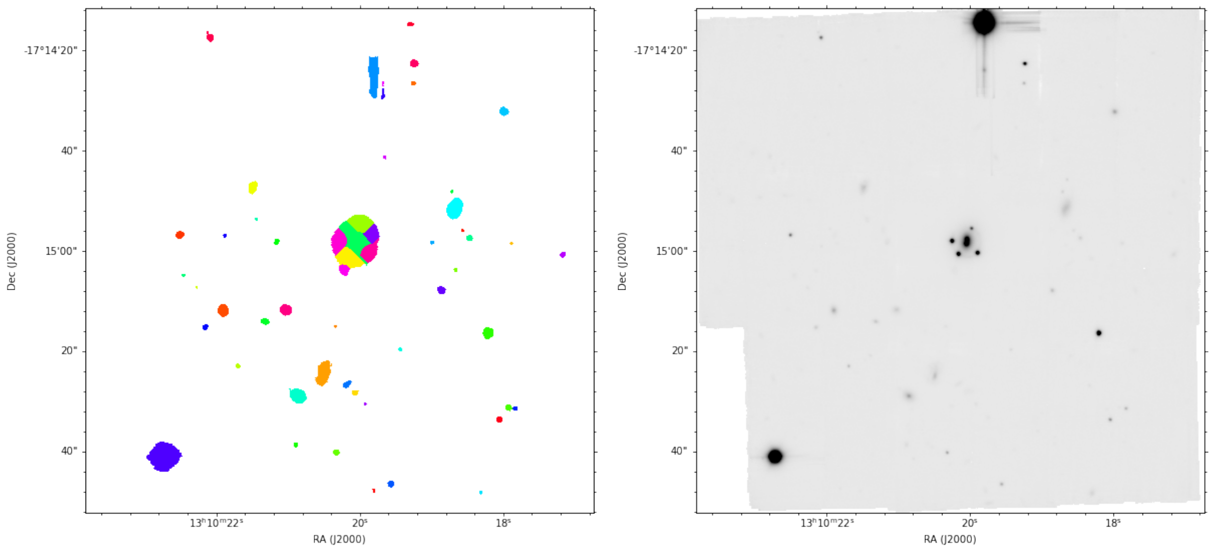

Figure C.4: Lensed quasar —MS1310— analyzed by AutoMUSE. There are 54 objects detected with detection threshold = 3. The target object seems to be over-segmented. There are 5 central sources, but 7 objects are detected in the segmentation map.