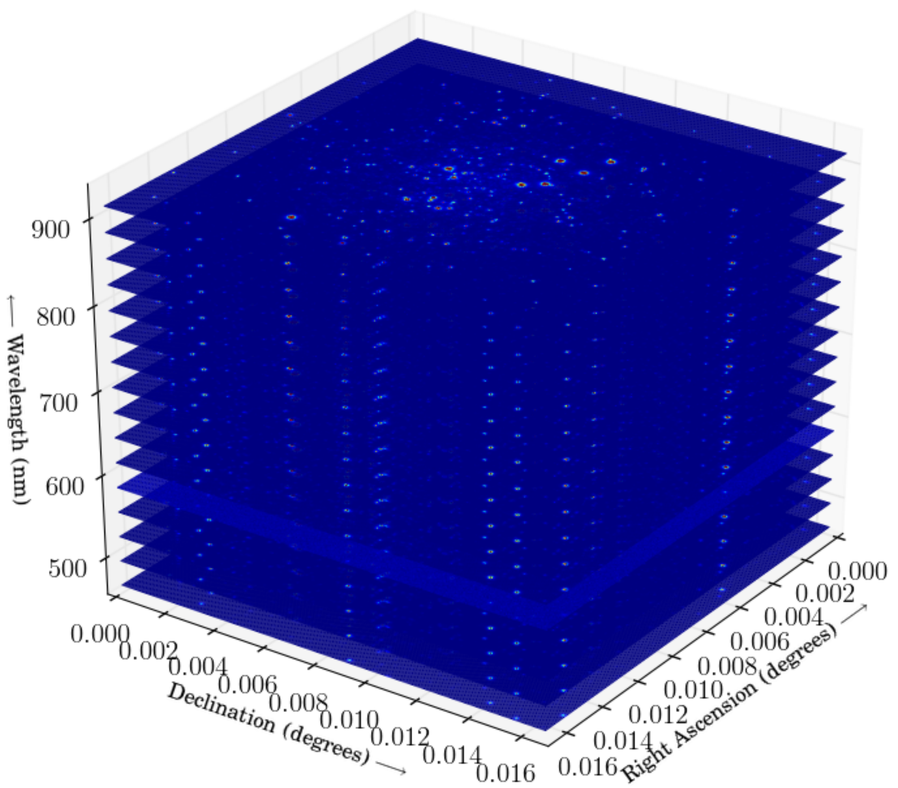
university of groningen
faculty of mathematics and natural sciences
kapteyn astronomical institute

Bachelor Research Project
"Resolved Stars in a large IFU Data Cube with GIPSY"

B. Hut
Supervisors: Prof. dr. E. Tolstoy & Dr. M.G.R. Vogelaar

4th April 2012



Abstract
The MUSE (Multi Unit Spectroscopic Explorer) is a second generation instrument to be installed on the VLT (Very Large Telescope) of the ESO (European Southern Observatory) on Paranal in Chile later this year. This new instrument produces two spatial coordinates (1× 1 arcmin) and one spectral coordinate (465.00 - 930.14 nm). This provides a lot of detailed information to solve astrophysical problems. The large data cubes are relatively new to optical astronomy while radio astronomy has used them for years. GIPSY (Groningen Image Processing System) is a well established example of software developed to analyse 3D data cubes from radio telescopes. Using this package in combination with Python on a simulated data cube of resolved stars in a globular cluster provided by the MUSE consortium, an algorithm to handle such a large optical data cube is developed during this research project. How one can extract spectra of individual stars, and measure the equivalent widths of the Ca II triplet absorption lines and use them to determine metallicities of individual stars is shown. This algorithm should be useful not only to handle simulated data but also real data which will soon arrive from MUSE.

Table of Contents

muse
multi unit spectroscopic explorer

GIPSY **Groningen Image Processing System**

Chapter 1

Introduction

The scheduled commissioning of the innovative Multi Unit Spectroscopic Explorer (MUSE) for ESO's Very Large Telescope (VLT) later this year will be an important moment in the study of galaxies. This new instrument contains 24 3D spectrographs and will be an efficient explorer of the Universe in three dimensions: 2 spatial coordinates and a spectral coordinate. MUSE is a unique tool to solve astrophysical problems.
Such one problem is our understanding of galactic evolution: how does the metallicity distribution of a stellar population change with time[5]? MUSE can be used to determine metallicities of individual stars in nearby resolved systems as well as integrated metallicities for more distant systems.

This report is divided into two parts: a description of the software required to analyse a simulated 3D MUSE data, and a practical application to measure the metallicities of stars from the CaII triplet lines. The first part will show how to pick out stars and their spectra using the Groningen Image Processing System (GIPSY). The second part shows how to turn these measurements into metallicities.
The instrumental and astronomical issues are combined in the research question:
What are the possibilities to process a 3D MUSE data cube with GIPSY and measure accurate metallicities using the CaII triplet lines?

1.1 What is a Data Cube?

To be able to answer the main question, some more detailed questions are first addressed: what does a 3D MUSE data cube look like and how is it composed? How does one deal with such a cube? How to do proper science with it?
A data cube is a representation of a file that consists of spaxels. A spaxel represents a measured intensity as function of (x, y, λ), just like a pixel is a function of (x, y). Figure 1.1 shows an illustration of a data cube.
First some astronomical aspects of metallicities of stars will be discussed and after that the way of retrieving information from a 3D MUSE data cube will be discussed.

1.2 MUSE

MUSE (the Multi Unit Spectrographic Explorer) is a new instrument which shortly will be commissioned on the VLT. It is an integral field unit (IFU) which can be used with the aid of adaptive optics. An IFU is a spectrograph that can measure spectra over a 2D area of the sky, which makes an IFU a 3D spectrograph. In the case of MUSE, there are 24 IFU's packed together to cover a large Field of View (FoV) which in its default 'Wide Field Mode' (WFM) is 1×1 arcmin, with

0.3 − 0.4 arcsec spaxels. In its 'Narrow Field Mode' (NFM), MUSE has a higher spatial resolution (0.030 − 0.050 arcsec spaxels) at the cost of FoV (7.5 .5 arcsec squared), see Table 1.3. The IFU's divide the FoV into smaller beams, which all enter a spectrograph. Table 1.3 shows the technical specification of MUSE and Figure 1.2 shows the 24 IFU's. The CCDs in the spectrographs contain $\sim 16 \cdot 10^6$ pixels and are cooled down to a temperature of about 140 K. In the end, the data of all the individual IFU's are combined into a 3D MUSE Data Cube. This data cube contains $\sim 370 \cdot 10^6$ pixels[21]. Such big files need to be handled by very efficient tools. This is exactly where GIPSY comes in.

1.3 GIPSY

The simulated 3D MUSE data cube that is used here contains two spatial axes (RA, DEC) and a spectral (λ) axis. This is a data type which is very common in radio astronomy, but still something of a novelty in optical astronomy. GIPSY is an interactive software system for the reduction and display of astronomical data developed in Groningen at the Kapteyn Astronomical Institute. GIPSY has mainly been used for radio data: in radio astronomy, the redshift is measured by the change in position of a single spectral line, usually the 21cm HI line. These measurements can subsequently be used to create a velocity field. GIPSY has its origins in the early 1970s when radio 21cm HI data typically contained 2 planes of 512×512 pixels or 32 planes of 128×128 pixels. That results in data cubes containing a total of 524288 data pixels.
Because GIPSY developers have had many interactions with users, GIPSY has always been evolving. New ideas in image analysis and user interaction have extended the functionality of GIPSY. In recent years for example, a set of highly interactive graphical user interface (GUI) components and a Python binding have been added. This Python binding provides a robust tool to analyse data cubes. Nowadays state-of-the-art data cubes like a MUSE 3D data cube, containing $370 \cdot 10^6$ data points, can be handled by GIPSY, using advanced techniques[3 and references therein]. This report investigates how to use GIPSY to extract spectra of stars in a simulated MUSE data cube and to measure the equivalent widths of the Calcium II Triplet lines to find the metallicities of individual stars.

	Wide Field Mode (WFM)	Narrow Field Mode (NFM)
Field of View	1×1 arcmin	7.5×7.5 arcsec
Spatial sampling	0.2×0.2 arcsec	0.025×0.025 arcsec
Spatial resolution (FWHM)	0.3 − 0.4 arcsec	0.030 − 0.050 arcsec
Spectral resolution	0.13 nm	0.13 nm

Table 1.1: Technical specifications of MUSE.

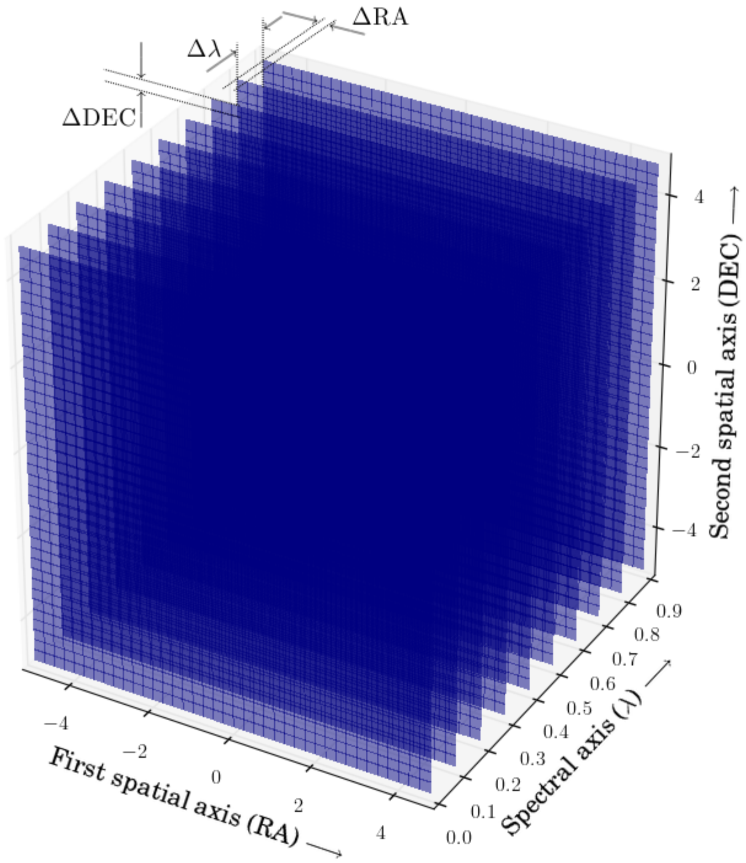


Figure 1.1: Illustration of a data cube.

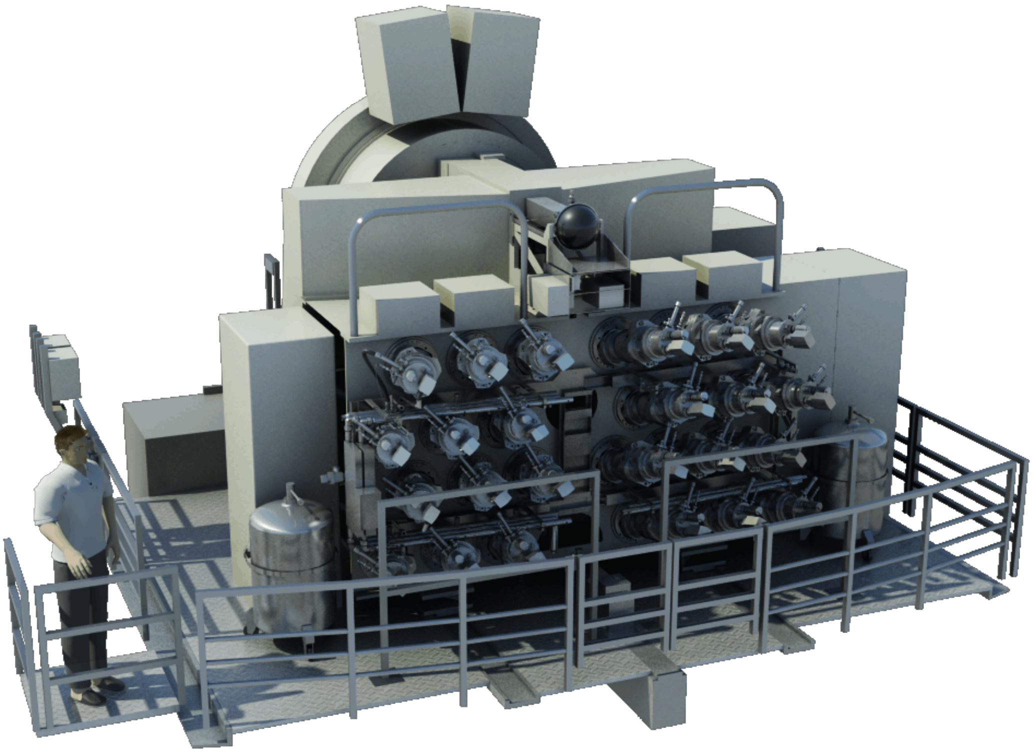


Figure 1.2: A prediction of how MUSE will look on the Nasymth platform of the VLT. The 24 individual IFUs can be clearly seen. The full FoV incident beam from the VLT will be split in 24 sub-FoVs. Each sub-FoV beam goes into an IFU where the spectra are recorded. The spectra of the IFU's will be combined into one big data cube[2].

Chapter 2

The Ca II Triplet

This chapter explains how the Ca II Triplet can be used as a metallicity indicator.

2.1 Introduction

The metallicity distribution function of a stellar population with time is an important aspect in understanding galaxy formation and evolution. The metallicity of a star is the amount of metals* it contains. It is defined to be the log of the ratio of iron to hydrogen in the star, divided by this ratio for the Sun, see equation 2.1. The metallicity is always relative to the Sun because absolute values are not precise[6]:

$$[\mathrm{Fe/H}] = \log_{10} \frac{(N_{\mathrm{Fe}}/N_{\mathrm{H}})_\star}{(N_{\mathrm{Fe}}/N_{\mathrm{H}})_\odot} = \log_{10} \left[\frac{N_{\mathrm{Fe}}}{N_{\mathrm{H}}}\right]_\star - \log_{10} \left[\frac{N_{\mathrm{Fe}}}{N_{\mathrm{H}}}\right]_\odot \tag{2.1}$$

The unit of [Fe/H] is dex, N_{H} and N_{Fe} are the amount of respectively hydrogen and iron.

2.2 Near infrared Calcium II Triplet

Detailed abundance analysis with high-resolution spectroscopy is time-consuming for large data sets containing numerous individual stars. Happily, there is an empirical method that can make an efficient estimate of [Fe/H] for individual red giant branch (RGB) stars using Ca II triplet (CaT) lines[2][5]. The upcoming sections are about this CaT and how the empirical method can turn equivalent widths of the CaT lines into [Fe/H]. As be will described in chapter 3, the MUSE spectral range covers the 3 lines of the CaT. These lines are due to transitions between the states $3^2\mathrm{D}$ and $4^2\mathrm{P}^0$. These lines are at rest, in vacuum at

$$\lambda_1 = 849.8 \text{ nm} \qquad \lambda_2 = 854.2 \text{ nm} \qquad \lambda_3 = 866.2 \text{ nm},$$

and they are by far the strongest absorption lines in the spectral region[18] (see figure 2.1a). Gaussian line depths ℓ_i are scaled in the ratios

$$\ell_1 : \ell_2 : \ell_3 \quad :: \quad 3 : 5 : 4. \tag{2.2}$$

2.3 From Equivalent Width to Metallicity

The characteristics of the CaT lines can be described in various ways. One of the most robust is the equivalent widths of each line. Here it is defined what this means and the relation of the equivalent width to the metallicity is explained.

*astronomical metals: elements heavier than helium

2.3.1 Definition of Equivalent Width EW

The equivalent width EW is a measure of line strength. It is defined to be the integral of the flux over wavelength:

$$EW \equiv \int 1 - \frac{F_\lambda}{F_0} d\lambda \tag{2.3}$$

Where F_λ is the flux at wavelength λ and F_0 is the flux of the continuum at the same wavelength. The equivalent width is defined as the shape that has one vertice at the zero-intensity line and two adjacent lines parallel to the intensity axis. The width of this shape with a surface that equals the surface between the spectral line and the continuum is the equivalent width, see Figure 2.2. In general, the equivalent width is denoted as EW, the equivalent width of the i^{th} spectral line is denoted by EW_i.

2.3.2 Definition of Reduced Equivalent Width W'

The dependence of a CaT line strength on metallicity is theoretically difficult to understand, yet it has been empirically proved by extensive calibration using RGB stars in globular clusters[5]. A linear combination of the CaT EWs depends on [Fe/H], the gravity g and the effective temperature T_{eff} of each star. As $\log g$ and T_{eff} decrease going up the RGB, the effect of gravity and temperature can be removed by taking into account the position of the star on the RGB with respect to the horizontal branch (HB)[22]. This is most efficiently achieved by defining a reduced equivalent width W':

$$\begin{aligned} W' &\equiv \sum W + \beta(V - V_{\text{HB}}) \\ &= EW_2 + EW_3 + 0.64(\pm 0.02)(V - V_{\text{HB}}) \end{aligned} \tag{2.4}$$

Where $\sum W$ is a linear combination of the individual line strengths, V is the stars absolute magnitude and V_{HB} is the mean absolute magnitude of the Horizontal Branch. Using V_{HB} also removes any strong dependence on the distance and/or reddening. The term $V - V_{\text{HB}}$ has an advantage over absolute magnitude or color, since then the slope β is constant. W' is thus the CaT line strength index at the level of the HB[5]. The linear combination and slope used is given as the second line of equation 2.4.

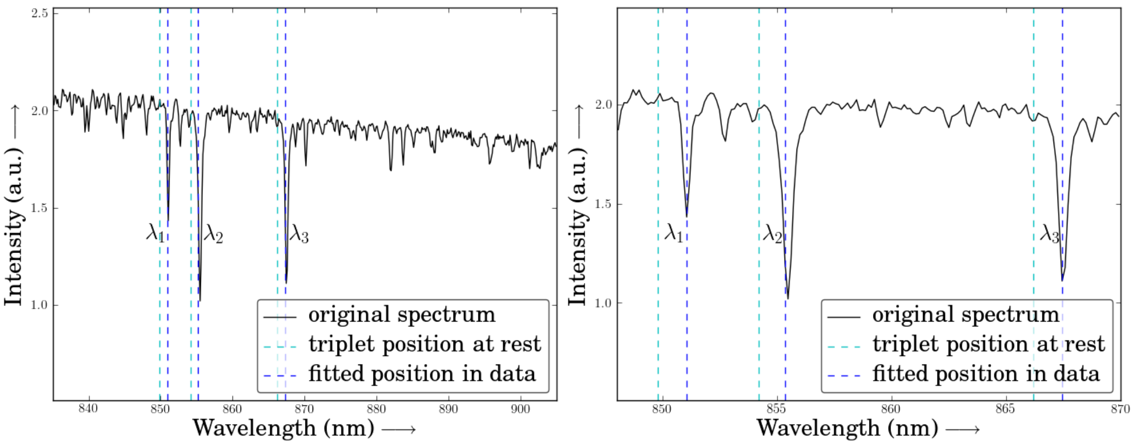
2.3.3 Relation between EW, W' and [Fe/H]

It has been shown that the CaT lines as indicators of [Fe/H] have advantages with respect to individual iron lines. CaT lines are very strong and in positioned in a spectral region with few other strong lines. The empirically derived correlation shows that the CaT lines of a metal rich RGB star are stronger than a metal poor RGB star of the same luminosity[4].

The equivalent widths can be turned into metallicities with the following empirical equation[6]:

$$[\text{Fe/H}] = -2.81(\pm 0.16) + 0.44(\pm 0.04)W' \qquad \text{for } -2.5 < [\text{Fe/H}] < -0.5 \tag{2.5}$$

Where [Fe/H] is an expression for the metallicity, defined in equation 2.1. W' is defined in equation 2.4. The numbers in this equation come from the work of Bettaglia et. al[5]. It is useful to note that equation 2.4 cannot hold for extremely low metallicities, since equations 2.4 and 2.5 imply a negative equivalent width. Negative equivalent widths are obviously physically meaningless. This basic empirical formula has been given a solid physical foundation[6].



(a) **(b)**

Figure 2.1: (a) A representative spectrum of the Near Infrared Calcium II Triplet of a single star in the simulated MUSE data. Note the shifted position of the CaT: this CaT is *not* at rest wavelength. The velocity shift is due to the motion of the star with respect to the Earth. **(b)** The same spectrum which is zoomed in on Ca II Triplet regime.

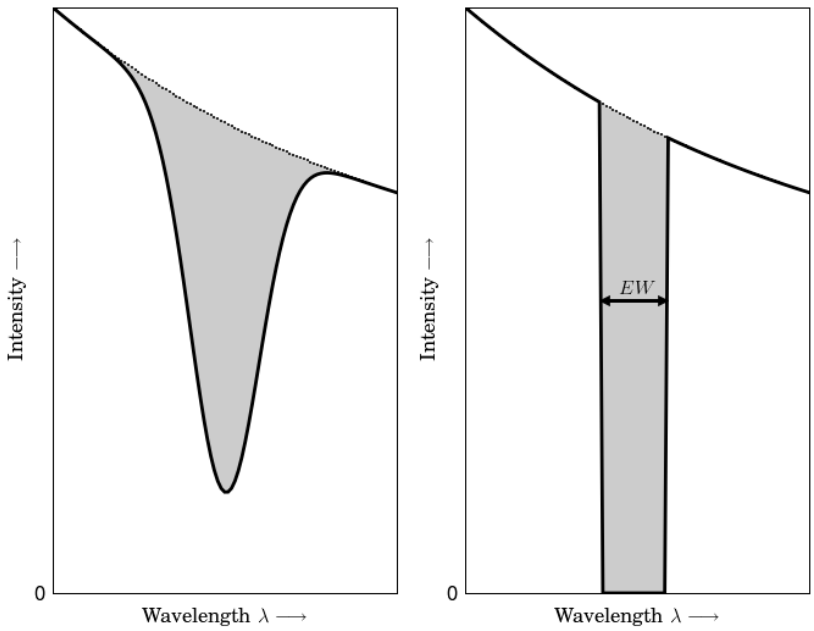


Figure 2.2: Graphical definition of equivalent width EW. The shaded surfaces have equal area.

Chapter 3

Simulated 3D MUSE Data Cube

The previous chapter explained how and why one would want to measure the equivalent widths of spectral lines and how to turn the CaT equivalent widths into a metallicity. In this chapter the analysis of a 3D MUSE data cube is discussed to obtain accurate CaT *EW*s. Useful results require that *EW*s are reliable and consistently measured. The first question is how to handle the data cube correctly with GIPSY. After this, a spatial analysis is carried out to locate the individual stars in the data cube, and the spaxels around each identified star are combined to create a spectrum. Finally, the flux of each star is summed over both spatial and spectral axes.

3.1 The Data Cube

A simulated 3D data cube of a MUSE observation of a globular cluster was provided by the MUSE instrument team. The wavelength range covers the near infrared CaT, which is an excellent metallicity indicator as shown in chapter 2[4][5][6] and can also be used to determine accurate radial velocities[5]. The data cube was provided as a FITS (Flexible Image Transport System) file, which is the standard archival data format for astronomical data sets[10]. There were some initial problems to put it into a standard format that GIPSY could read. This was mainly caused by FITS header keywords that are not supported by GIPSY. For example, inspection of the header showed that the intensity values did not have a unit (see appendix A). The intensity unit was assumed to be erg $s^{-1}cm^{-2}$. The header values are given in Table 3.3.1.

3.2 Preparation of the File

The 3D MUSE data cube is more complex than the HI data cubes normally seen by GIPSY[3]. This is because the amount of information is much more dense. The 3D MUSE data cube is converted into a GIPSY readable format by the new GIPSY task `FITSREPROJ`. `FITSREPROJ` creates a classic GIPSY header while the image data is simply copied and thus not affected. See appendix A for the usage of this task. The counts in the image data are of the order of magnitude 10^{-16} and it is therefore easier to rescale by a factor 10^{16} before doing any calculations. Afterwards the data is rescaled again to the original values. Once the data cube is readable by GIPSY, the spaxels can be analysed.

3.3 Spatial Analysis

In order to make a spectral analysis of a 3D data cube of a globular cluster, first the individual point sources that are stars have to be identified. Figure 3.1 shows an image at $\lambda = 850.0$ nm and figure 3.2 shows a subsection for both $\lambda = 850.0$ nm and $\lambda = 465.0$ nm. For the plotted stars, it is clear that the peaks are higher at $\lambda = 465.0$ nm. To analyse spectra, it is convenient to use a subsection as small as possible: see figure 3.3. This will be discussed in section 3.4.

3.3.1 Finding Stars

First of all, the stars in the data cube have to be identified. Finding stars can be done in multiple ways. The simplest way of selecting stars, is by clipping: stars that are brighter than a specific intensity value will be selected. Clipping has the advantage that it is a fast algorithm, as it selects all pixels higher than a certain threshold. This way, clipping can select multiple spaxels per star. In order to be more precise, one can look for patterns in x and y, where a star should have a peak within a certain range Δx and Δy.
In the case of this analysis of the MUSE data cube, first, the x direction is searched for a peak-shape in intensity. Where there is a candidate-star identified in the x direction, the same kind of shape should be present in y direction. If that is true, then the clipping method makes sure that the star is significantly brighter than the background noise level. In the end there is only one central spaxel per star.
The algorithm selects the position of stars based upon the shape of a star in both x- and y directions and the fact that it should have a higher signal than the environment. The script uses the sign of the derivative along the x- and y axis. It is assumed that the central pixel of the star is the brightest, the transition of a positive to negative slope will occur at exactly the coordinate of the star. Moreover, a clipping is done at 2% of the highest intensity value present. This sets the faint limit. The 2% limit is chosen as it is just above the standard deviation of the noise that is in the image, see figure 3.4. The result of this way of finding stars is plotted in figure 3.5, overlaying the image of figure 3.1 so that comparison will be possible. This algorithm is using GIPSY's python binding, because the tasks of GIPSY used to localize stars were not able to handle the very large MUSE data cube without preprocessing.

keyword	value	description
SIMPLE	T	conform to FITS standard
BITPIX	-32	array data type
NAXIS	3	number of array dimensions
NAXIS1	301	number of data points along RA axis
NAXIS2	301	number of data points along DEC axis
NAXIS3	3578	number of data points along spectral axis
EXTEND	T	
CDELT1	$5.555555555555556 \cdot 10^{-5}$	step right ascension
CDELT2	$5.555555555555556 \cdot 10^{-5}$	step declination
CDELT3	0.13	step wavelength
GCOUNT	1	number of groups
EQUINOX	2000	standard FK5 (years)
CUNIT1	deg	spatial unit of RA axis
CUNIT2	deg	spatial unit of DEC axis
CUNIT3	nm	unit of spectral axis
CROTA2	0.0	Appended by Kapteyn Package module Maputils 17d
CRPIX1	150.0	Start x spatial coordinate in pixel
CRPIX2	150.0	Start y spatial coordinate in pixel
CRPIX3	1.0	Start wavelength coordinate in pixel
CRVAL1	0.0	start RA spatial coordinate
CRVAL2	0.0	start DEC spatial coordinate
CRVAL3	465.0	start wavelength coordinate
PCOUNT	0	number of parameters
CTYPE1	RA—TAN	first axis is right ascension
CTYPE2	DEC–TAN	second axis is declination
CTYPE3	AWAV	third axis is wavelength

Table 3.1: Header of the MUSE data cube after processing by `FITSREPROJ`: the 3 axis of the cube are characterized by the number of data points along an axis (NAXIS), the step size between 2 data points along an axis (CDELT), the value of the first data point along an axis (CRVAL) and by the corresponding units (CUNIT). These quantities are illustrated in figure 1.1.

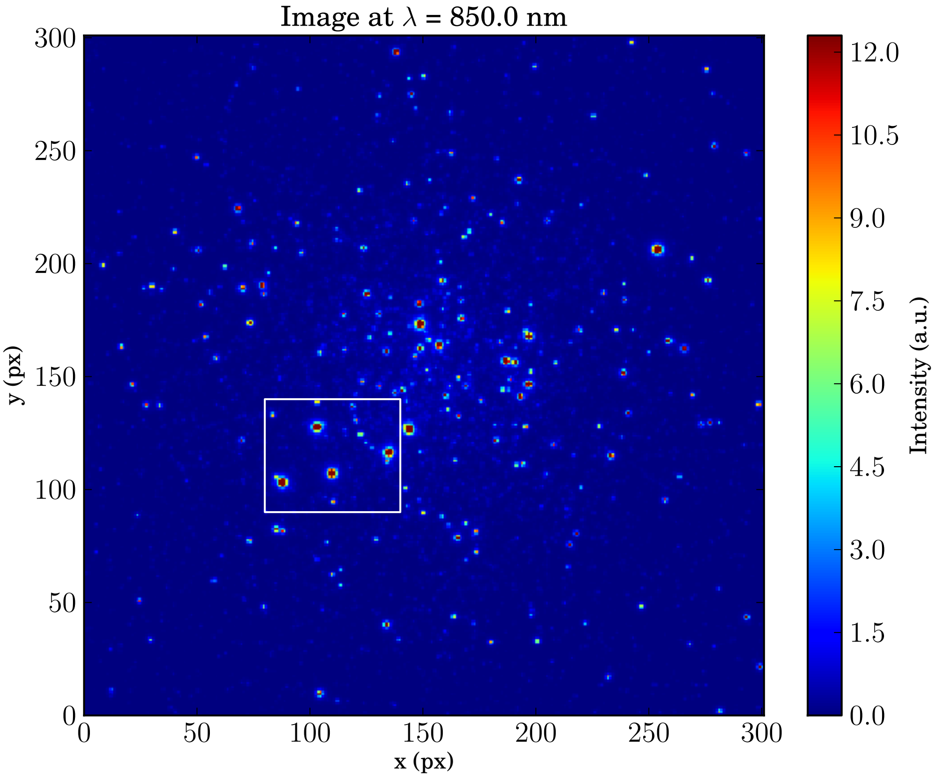


Figure 3.1: The full area of the sky of the 3D MUSE data cube at λ = 850.0 nm. See Figure 3.2 for zoom in of the white box.

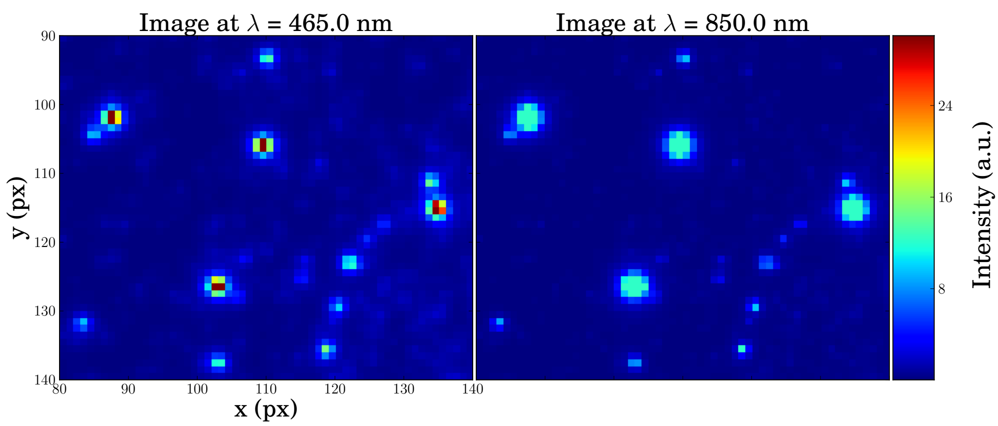
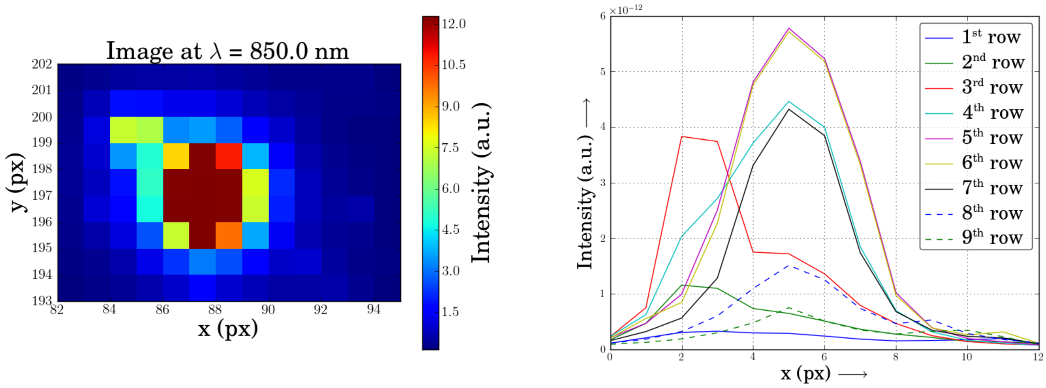


Figure 3.2: Zoom in of 4 bright stars in the white box in Figure 3.1 at *(left)* λ = 465.0 nm and *(right)* λ = 850.0 nm.



(a) (b)

Figure 3.3: **(a)** Zoom in on lower left star of Figure 3.2. **(b)** Profiles of intensity per row in image (a): in solid blue the intensity of the upper row, $y = 202px$.

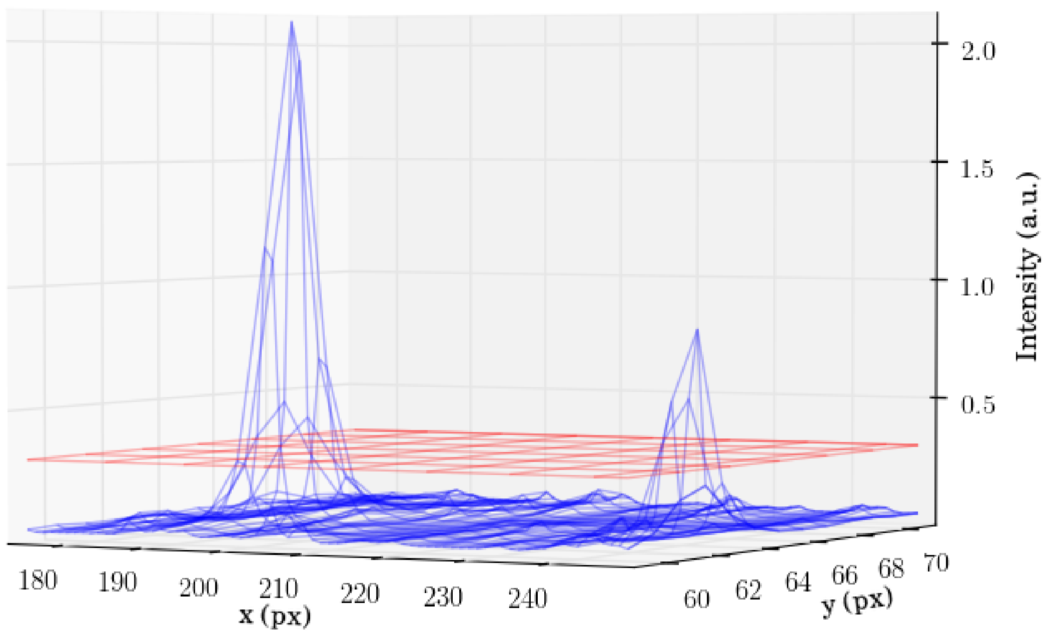


Figure 3.4: A visualisation of how the star finding algorithm works. The blue lines represent the pixel value in the x-y plane at λ = 465.0 nm and the red plane represents the clipping level. The algorithm finds 2 stars in this area.

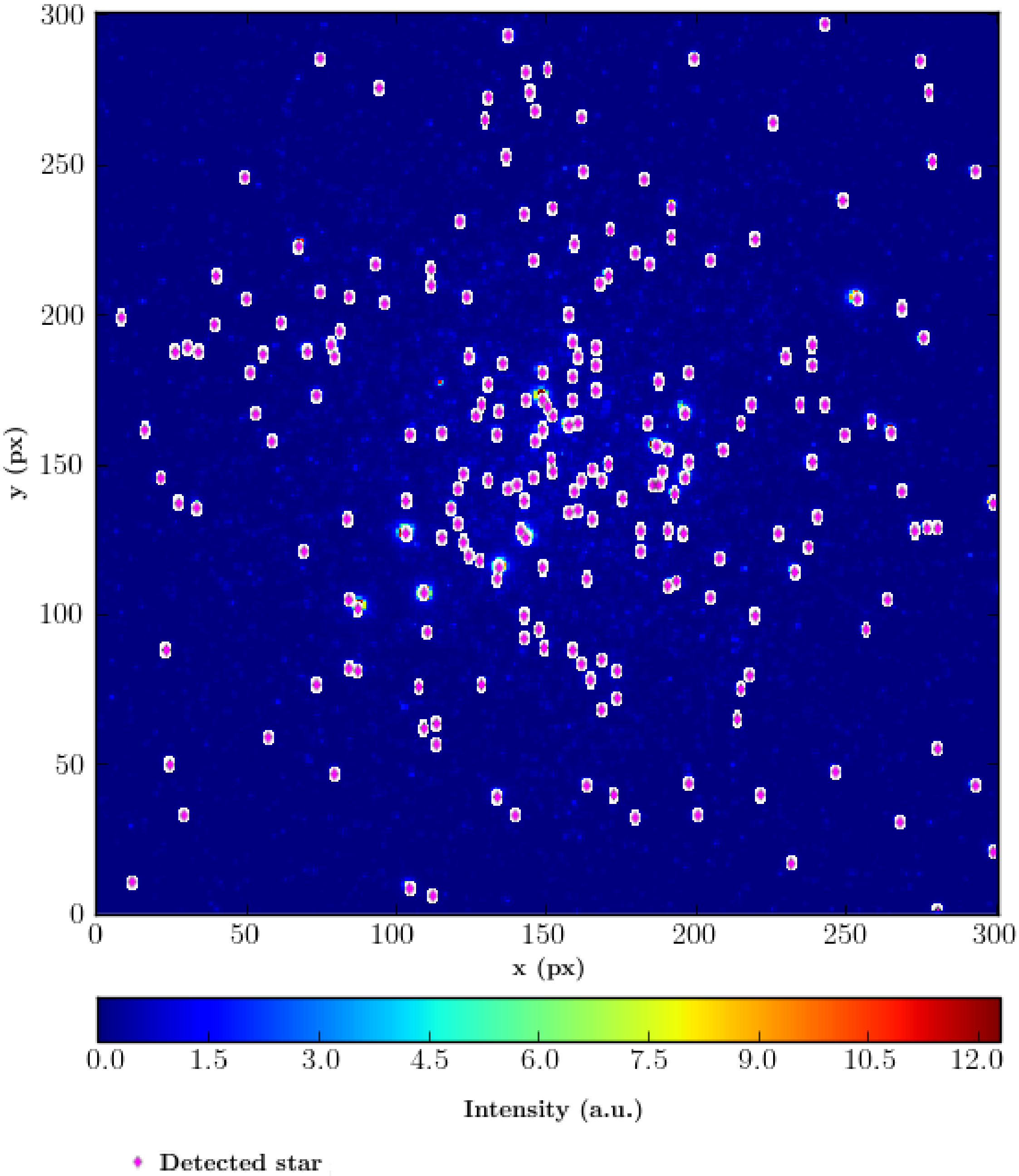


Figure 3.5: The central positions (x, y) of stars that are found in the simulated data cube. A total of 221 stars were found.

3.4 Spectral Analysis

In Figure 3.6 four spectra in the spectral range that MUSE covers of different spaxels around a single star are shown. The slope of the continuum and the existence of noise depends on the location of the spaxel. To analyse the CaT a wavelength range around the CaT is selected. In this range, the slope of the continuum is assumed to be constant. Figure 3.7 shows a zoom in on the spectral region around the CaT. For the star, the spectrum of the inner most brightest spaxels does not contain a CaT, unlike the less bright pixels which do show it. The reason for this is not certain, but it is thought to be saturation or some problem in the construction of the data cube. More information about saturation normally would be in the header, but that is not available for this simulated data cube.
From Figure 3.6, it is obvious the CaT is not present in the inner brightest spaxels. For intermediate bright spaxels the CaT is clearly visible and for outer spaxels noise becomes increasingly important. The fact that it is in general not possible to determine characteristics from the most bright spaxel, is another reason to sum all spaxels of a star, omitting the central 'saturated' spaxels. It is important to select the correct spaxels. There are multiple ways to select all spaxels that belong to a star. For example, the clipping method adds all spaxels that are intense enough and connected to the central spaxel. Another way is putting a small box around the central spaxel and sum the flux of all spaxels that are in the box. By increasing the size of the box, the signal to noise ratio SNR in the summed spectra should rise if the added spaxels still contains flux of the star. If not, the SNR will drop, because the added spaxels only contain noise (see Figure 3.8). Preferably, the shape of this box should be an ellipse, to match the telescopes point spread function. This report will use the last algorithm, since it was found to be most robust. Figure 3.9 shows how the number of pixels influences the summed spectrum. Initially a small box has been chosen. When the box size increases, the SNR rises. When the box is larger than the size of a star, only noise will be added and therefore the SNR drops again. The boxsize for summing flux is such, that the SNR has a maximum at that size.

3.5 Determination of Properties of each Star

In order to analyse each star, the spaxels are summed as described in the previous sections. Now it is known how to extract the spectra of the stars. Afterwards it will shown how the spaxels are also used to determine the properties of each star.

3.5.1 Fitting the Spectra

Each stellar spectrum from Section 3.4 has to be fitted by a theoretical curve that represents the shape of the CaT lines and the continuum. It is assumed that the absorption lines are three normal distributed line functions, representing the CaT and that the continuum is a linear offset :

$$\begin{aligned} I_T(\lambda) &= \sum_{\text{all triplet lines}} I_{\text{line}}(\lambda) + I_{\text{continuum}}(\lambda) \\ &= \sum_{i=1}^{i=3} A_i \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(\lambda_i-\lambda)^2}{2\sigma_i^2}} + (a\lambda + b). \end{aligned}$$

Where A_i is the amplitude of the i^{th} triplet line, σ_i is the standard deviation of the gaussian triplet line and λ_i is the position of the gaussian triplet line. a is the slope of the continuum and b is its offset. To reduce the number of parameters that needs to be fit, the known gaussian line depths ℓ_i of each line are assumed. For the CaT, $(\ell_1 : \ell_2 : \ell_3) = (3 : 5 : 4)$. Now A_i is replaced by an overall amplitude A_0. The amplitude of the individual lines is $\ell_i A_0$. Furthermore, in the last step the known relative positions of the guassian lines are also inserted. d_i denotes the spectral distance between λ_0 and the position of the i^{th} line. For the CaT, $d_1 = \lambda_1 - \lambda_0 = 0$ nm,

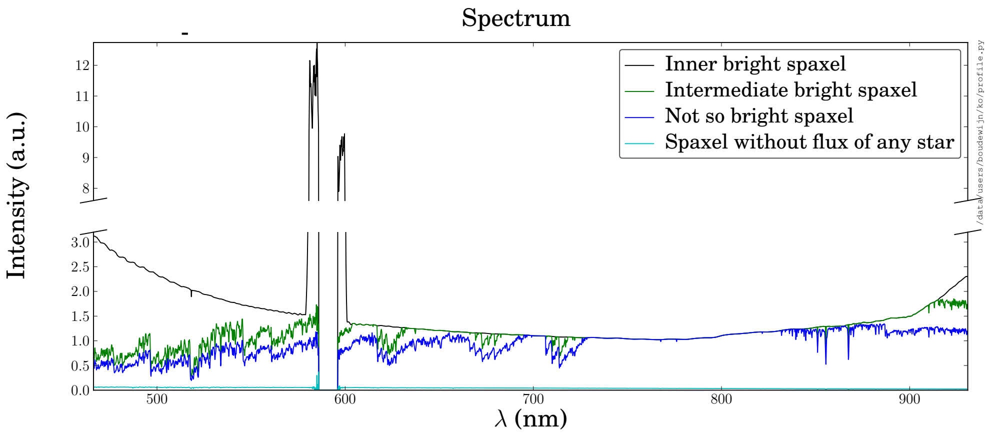


Figure 3.6: The complete spectrum from MUSE for a very bright star of one inner bright spaxel (black), an intermediate bright spaxel (green), an outer spaxel (blue) and a spaxel that does not contain flux of a star (light blue). The spectrum in the range around the CaT is plotted in Figure 3.7.

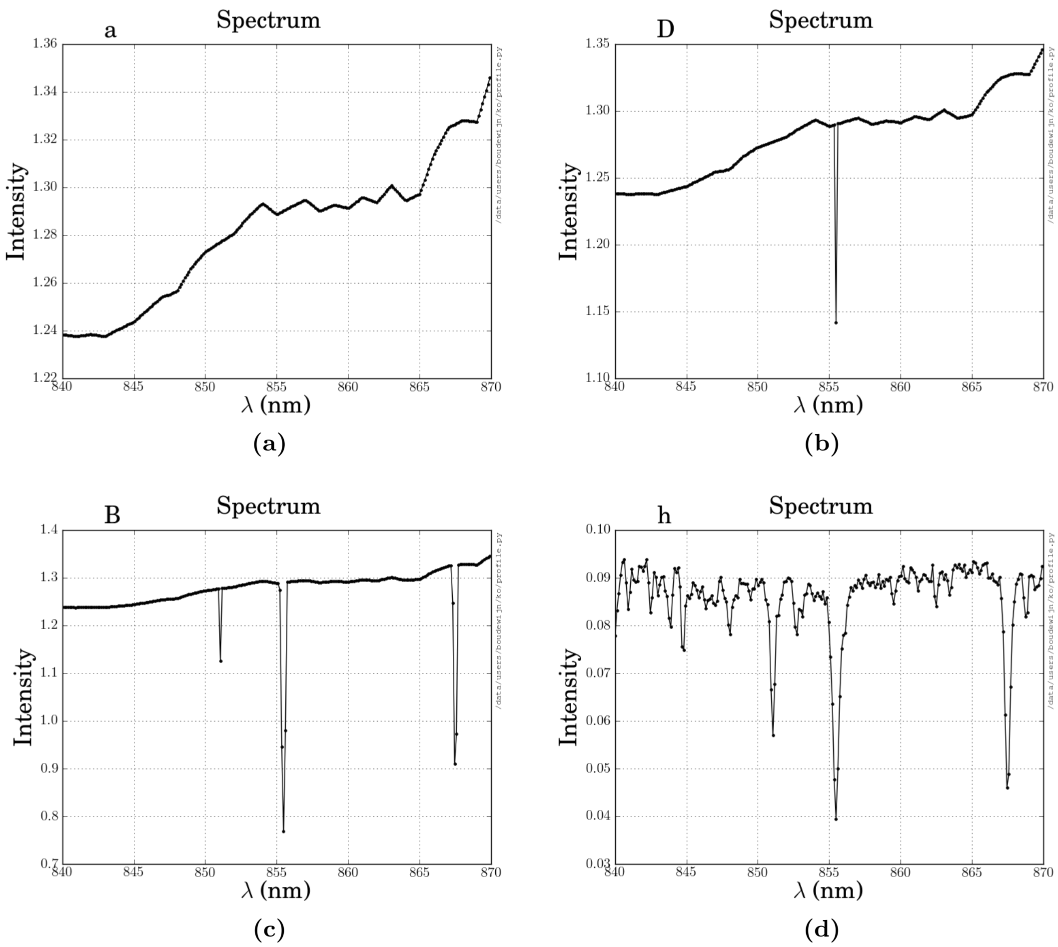


Figure 3.7: Spectra around the CaT triplet wavelength for a very bright star of two inner **(a)** & **(b)** bright spaxels; **(c)** an intermediate bright spaxel; **(d)** an outer, not so bright spaxel.

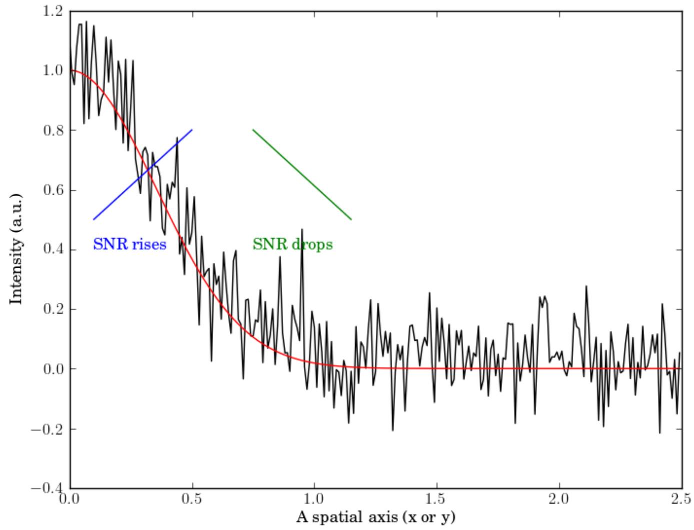
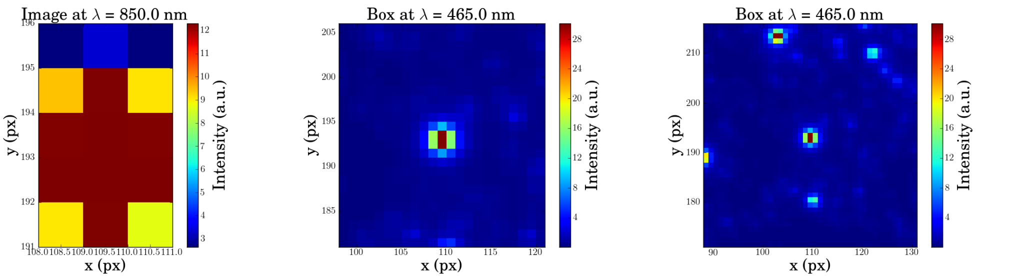
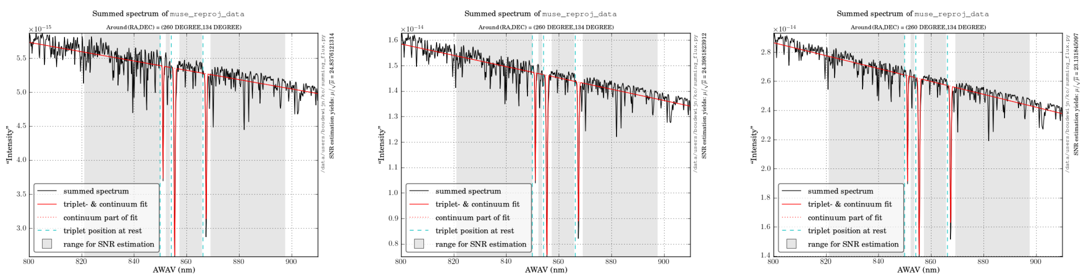


Figure 3.8: Flux summing algorithm: Intensity along one spatial axis are shown where the red line shows a gaussian star. Increasing the area to sum the flux, will increase SNR for a short interval. Adding flux 'outside' the star will worsen the SNR.



(a) Width = 3px; Height = 5px **(b)** Width = 23px; Height = 25px **(c)** Width = 43px; Height = 45px



(d) SNR = 17.3 **(e)** SNR = 31.2 **(f)** SNR = 28.2

Figure 3.9: **(a)**, **(b)** and **(c)** shows the box used for summing flux for increasing box sizes. **(d)**, **(e)** and **(f)** shows the corresponding summed flux. More of such figures are in appendix B.2.

$d_2 = \lambda_2 - \lambda_1 = 4.4$ nm and $d_3 = \lambda_3 - \lambda_1 = 16.4$ nm.

$$\begin{aligned} I_T(\lambda) &= \sum_{i=1}^{i=3} \ell_i A_0 \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(\lambda_i-\lambda)^2}{2\sigma_i^2}} + (a\lambda + b) \\ &= \sum_{i=1}^{i=3} \ell_i A_0 \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{((\lambda_0+d_i)-\lambda)^2}{2\sigma_i^2}} + (a\lambda + b). \end{aligned} \tag{3.1}$$

Equation 3.1 is used in the fitting procedure. The parameters that should be fitted are: the overall amplitude A_0 and position λ_0, the individual line's σ_i, and the slope a and offset b of the continuum. To start the fitting procedure, it is necessary to make initial estimates for the unknowns.

Initial estimates of the continuum

To make an initial estimate of the continuum, the least square method is applied to a straight line[14]. For a straight line $y = ax + b$, this analytical method yields:

$$a = \frac{m\sum_{i=1}^m x_i y_i - \sum_{i=1}^m x_i \sum_{i=1}^m y_i}{m\left(\sum_{i=1}^m x_i^2\right) - \left(\sum_{i=1}^m x_i\right)^2} \quad \text{and} \quad b = \frac{\sum_{i=1}^m x_i^2 \sum_{i=1}^m y_i - \sum_{i=1}^m x_i y_i \sum_{i=1}^m x_i}{m\left(\sum_{i=1}^m x_i^2\right) - \left(\sum_{i=1}^m x_i\right)^2} \tag{3.2}$$

Where m is the number of data points in the data set.

Initial estimates of the CaT

The initial estimates of the overall amplitude A_0 of the triplet and its position λ_0 are estimated using the known gaussian line depths ℓ_i and the distance between the lines d_i that tell what the theoretical CaT should look like. The distance d_i is used to select three data points in the spectrum. For these points, it is checked that the intensities follow the expected gaussian line depths up to a certain tolerance. If the three data points do not follow this, three new data points are selected. The selection starts from the position of the first line at rest and iterates to higher wavelength. If the data points follow the expected gaussian line depths, the depth with respect to the continuum and the position of those data points are used as initial estimate for the fitting procedure of the function in Equation 3.1. Finally, an initial estimate of σ_i is made. A width of 0.75 nm is used for summing the flux of a CaT line, showing that $\sigma_i \leq 1.5$ nm[5]. By eye, the initial estimate is manually set to 0.3 nm (see Figure 2.1b).

3.5.2 Determining the metallicity

Having a good fit of the CaT and its individual lines makes it possible to determine the position of the CaT, the equivalent widths of the lines and the signal-to-noise. Using these, the metallicities and velocities can be determined for each star and the measurement errors. The equivalent widths per line can be determined, using equation 2.3.
Because the magnitude or colour values of the stars in the data cube were not given, they were estimated. This is done using the relative levels of the continuum flux at the same position in each spectrum and also at a different position to get a measure of colour. These estimates are rough but sufficient for the purposes in this research project. These flux measurements can be turned into an apparent magnitude V_{app} using:

$$V_{\text{app}} = -2.5\log_{10}(\text{flux}) + V_0 \tag{3.3}$$

where V_0 is a zero point magnitude. Using the distance modulus μ, this becomes an absolute magnitude V:

$$\begin{aligned} V &= V_{\text{app}} - \mu \qquad (3.4) \\ (\mu &= 5\left[\log_{10}(d)\right] \qquad \text{distance } d \text{ in pc}) \end{aligned}$$

In the CaT method the difference in magnitude of the horizontal branch V_{HB} and the observed star is used. Combining V, V_{HB} and the EWs according equation 2.4, the value for W' is determined, and using equation 2.5 this becomes a metallicity, [Fe/H].

Signal-to-noise estimation

The signal-to-noise ratio (SNR) is a measurement of the fluctuations in the spectrum compared to the signal. It can be computed from a line free wavelength range, where the signal is due to the continuum, comparing this to the noise fluctuations in the same regime. It should be correlated with the magnitude of the star. For the same exposure time, the noise is constant and the brighter the star, the more signal and thus the higher SNR. In this project, the signal-to-noise is photon-limited because the faint-object observations are done using a CCD detector[15]. The relevant equation in this case is,

$$\mathrm{SNR} = \frac{S}{N} = \frac{\mu'}{\sqrt{\mu'}} \qquad \text{with } \mu' \text{ is the mean signal.} \tag{3.5}$$

3.6 Error Analysis

In order to make an estimate of the error in the result that is given by the algorithm of the previous sections, the error in equivalent width will be calculated via error propagation. First, the error in the intensity of the fit will be considered and that will be used to estimate the error in the equivalent width of each line. Afterwards the error in other measurements will be considered.

3.6.1 Error in EW

Per CaT line it is assumed that the error of A_0 and σ_i are independent. Then, one can add the errors quadratically to get the error in line intensity:

$$\begin{aligned}
\Delta^2_{I_{\mathrm{line}\ i}} &= \left(\frac{\partial I_T}{\partial A_0}\right)^2 (\Delta_{A_0})^2 + \left(\frac{\partial I_T}{\partial \sigma_i}\right)^2 (\Delta_{\sigma_i})^2 \\
&= \left(\sum_{i=1}^{i=3} \ell_i \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{((\lambda_0+d_i)-\lambda)^2}{2\sigma_i^2}}\right)^2 (\Delta_A)^2 \\
&+ \left(\sum_{i=1}^{i=3} \ell_i A_0 \frac{2d_i^2 - 4d_i(\lambda_0-\lambda) + 2\lambda_0^2 - 4\lambda_0\lambda - \sigma_i^2 + 2\lambda^2)}{2\sqrt{2\pi}\sigma_i^{7/2}} e^{-\frac{((\lambda_0+d_i)-\lambda)^2}{2\sigma_i^2}}\right)^2 (\Delta_{\sigma_i})^2
\end{aligned} \tag{3.6}$$

Δ is used here for the standard deviation, to make a distinction between the standard deviation σ of the gaussian line.
The value for Δ_A and σ_i will be the ones that are given by the standard deviation in these parameters from the fitting procedure.
The equivalent width is calculated by equation 2.3. To get the error ΔEW, the error in line intensity is used as a bound. From the maximum $EW_{\max}$ and minimum $EW_{\min}$ from the intensity $I_{\mathrm{line}i} \pm \Delta_{I_{\mathrm{line}i}}$ ΔEW is calculated as the average of both:

$$\Delta EW_i = \frac{|EW_{i,\max} - EW_i| + |EW_{i,\min} - EW_i|}{2}.$$

3.6.2 Error in other measurements

It is assumed that the algorithm finds the central position of each star. It may be that the actual position of the star is in a spaxel close the the central position. This error is estimated as the size of one resolution element, just as the error in the position of the triplet in the spectra:

$$\Delta x = \Delta y = 1\mathrm{px} \Rightarrow \Delta\mathrm{RA} = \Delta\mathrm{DEC} \simeq 5.55556 \cdot 10^{-5}\mathrm{deg} \quad \text{and} \quad \Delta\lambda = 0.13\mathrm{nm}.$$

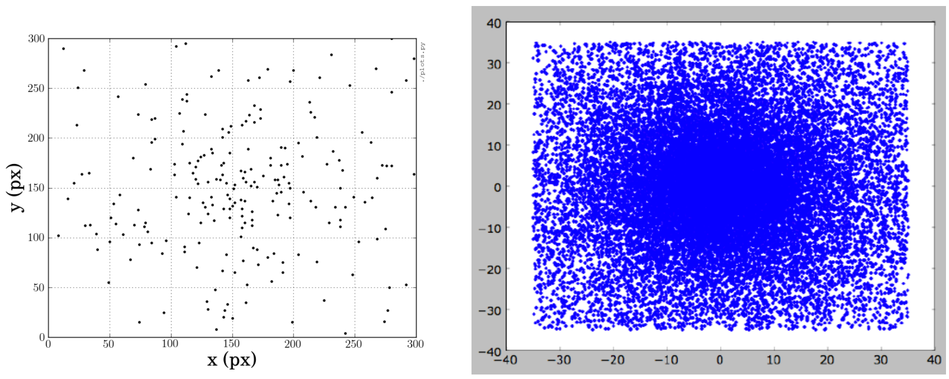
Chapter 4

Discussion of the Result

In this chapter the results of the previous chapter are analysed. The results for each star are shown in Appendix C. They are compared with what is known about the inputs. Since the MUSE instrumental team did not provide detailed photometry of the stars in the simulation, they are estimated[23]. At last, the fitting procedure is discussed using statistics.

4.1 x-, y Positions

In the simulated 3D MUSE data cube, the algorithm that is made for this report has detected 221 stars. This detection is based upon the shape of a star and a clipping level. In the simulated data cube, there are 23337 stars mostly below the detection threshold, to give realistic background noise with 1549 different spectra[23], see Figure 4.1. Moreover, the difference in number of detected stars can be influenced by the clipping level. Lowering the clipping level will make it possible to detect more stars. Furthermore, from figure 4.1b one can conclude that the realistic noise should be high due to the high density of stars and thus in the image crowding.

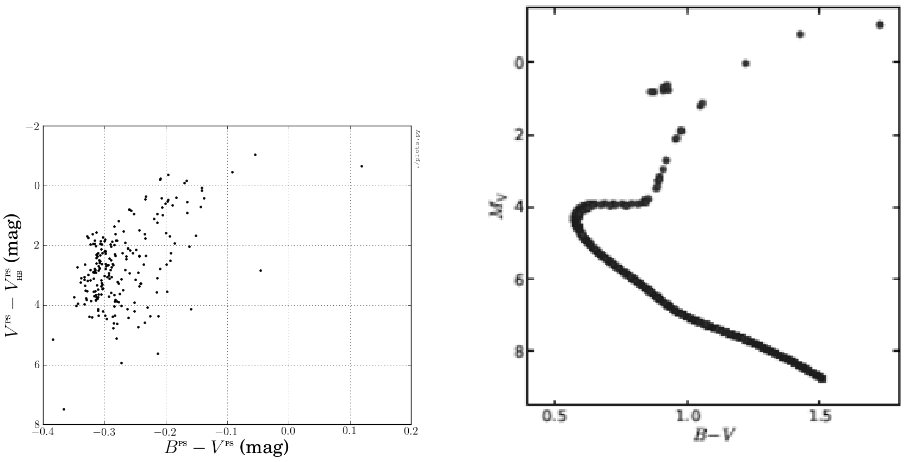


(a) (b)

Figure 4.1: x-y position diagram of **(a)** detected stars during this research. **(b)** Used by MUSE instrument team to simulation data cube[23].

4.2 Pseudo Colour-Magnitude Diagram

A pseudo colour B^{PS} and V^{PS} is determined for each star by simply using flux in a long wavelength interval for V^{PS} and flux in a short wavelength interval for B^{PS}. From a colour-magnitude diagram a main sequence and turn off should be visible. Since it is known that the simulated globular cluster only has 4 RGB stars (see figure 4.2b), Red Giant Branch stars should be visible. Recall from chapter 2 that the CaT method is calibrated using RGB stars, so the metallicities of those stars should be typical. $V_{\mathrm{HB}}^{\mathrm{PS}}$ is estimated to be 16.9 mag so that $V^{\mathrm{PS}} - V_{\mathrm{HB}}^{\mathrm{PS}}$ of the horizontal branch is about zero. All stars under $V^{\mathrm{PS}} - V_{\mathrm{HB}}^{\mathrm{PS}} = 0$ should be on the main sequence. The pseudo colour-magnitude diagram is shown in Figure 4.2a. Comparing the colour-magnitude diagram with the one that is used by the MUSE instrumental team (see Figure 4.2b), it is clear that it is hard to localize the turn off point in the results from this research. Localizing a Red Giant Branch is possible and the 8 most brightest stars are selected so the metallicities can be checked.

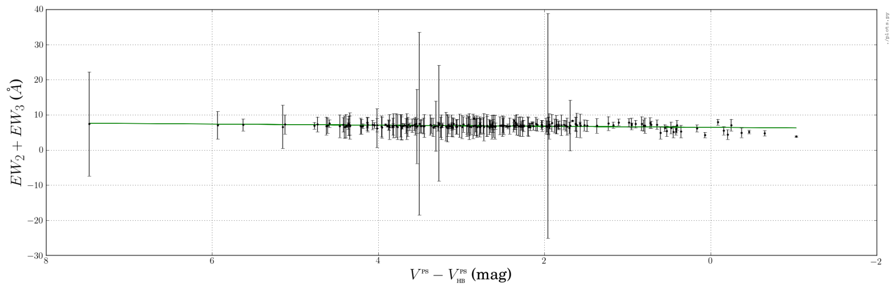


(a) (b)

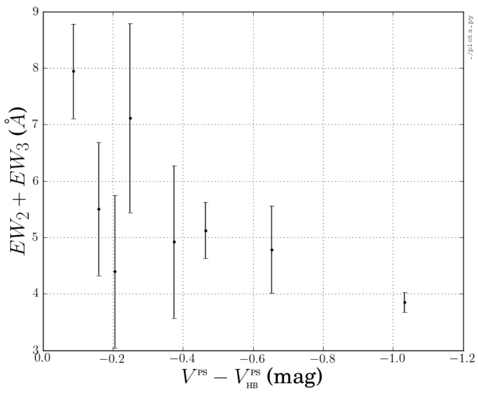
Figure 4.2: Pseudo color-magnitude diagram. **(a)** Measured with the algorithm of this research, using pseudo colours. **(b)** Of the simulated stellar population[23].

4.3 Metallicities

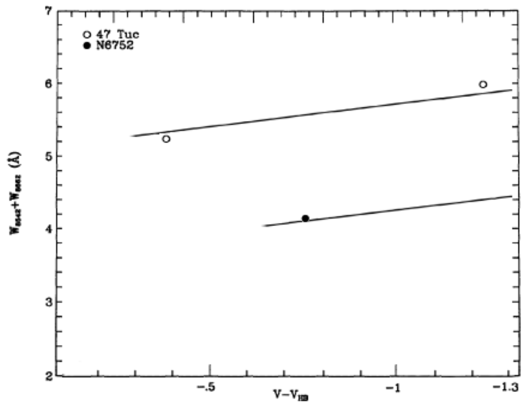
Some quantities of the stars that may be the RGB stars because of their position in the colour-magnitude diagram, are listed in Table 4.1. In the literature the summed pseudo-equivalent widths are often plotted to inspect cluster sequences. A cluster sequence is a linear line in a ($V^{\mathrm{PS}} - V_{\mathrm{HB}}^{\mathrm{PS}}$, $EW_2 + EW_3$) diagram[4]. To check the cluster sequence, Figure 4.3 can be used. Figure 4.3b shows the position for 8 brightest stars. It is clear that the slope of the cluster sequence would be negative, while it should be positive. This difference cannot be due to the fact that there are 8 stars plotted while there should only be 4 RGB stars. Selecting only the 4 brightest stars would still have a negative trend. Another explanation may be that the simulated data cube has same problems. This may be related to the spaxels with different properties described in section 3.4. This couldalso be related to the uncertainties in magnitude of the stars.It is nice to note that the order of magnitude of the pseudo-equivalent widths agrees the magnitude one would find in the literature. The magnitude of the metallicities in Table 4.1 should be ~ -0.7, since that is the value that is used to construct the data cube[23].



(a)



(b)



(c)

Figure 4.3: Summed reduced-equivalent width of the two strongest CaT lines against the pseudo magnitude difference $V - V_{HB}$. **(a)** Measured with the algorithm of this research for 222 stars. **(b)** Zoom of the most bright, including RGB stars. **(c)** Typical values for such a measurement in the corresponding $V - V_{HB}$ range for RGB stars[4].

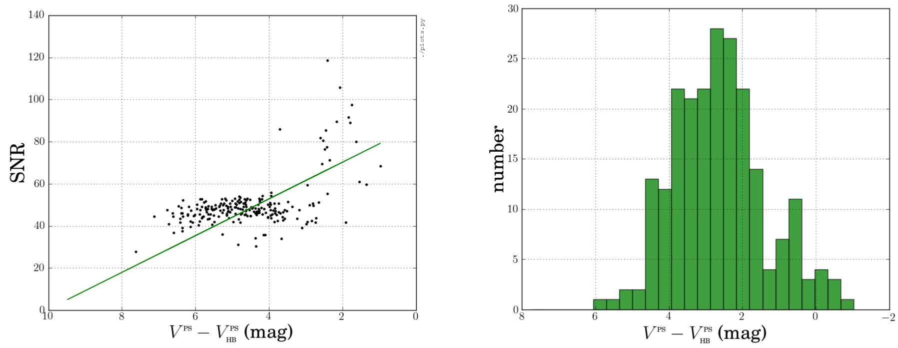
num	x	y	λ	EW_1	EW_2	EW_3	W'
	px	px	nm	nm	nm	nm	nm
65	124	115	851.069496035	3.023325	2.73241147226	2.76966826944	5.3997961935
76	114	124	851.064669558	111.6293	1.92054452734	1.9282049421	3.1880524247
95	183	137	851.051792771	2.755792	2.40271686748	2.3810104516	4.3656810553
97	157	138	851.069885401	2.566248	2.17741852789	2.21640974064	4.2619711100
128	137	159	851.06242146	2.879378	2.5532108356	2.57072194604	4.8267942463
129	159	160	851.07147618	2.728848	2.43885026307	2.47927218233	4.6784365635
169	190	191	851.069979652	3.519347	3.9420173636	3.99667302034	7.8826590724
217	298	280	851.069881536	3.740228	3.52329551786	3.587893706489	6.9520585054

num	SNR	B^{PS}	V^{PS}	$V^{PS} - V^{PS}_{HB}$	[Fe/H]
		mag	mag	mag	dex
65	91.5024034525	16.5735610822	16.740181956	-0.1598180440	-0.434089674839
76	68.5120009039	15.8129063502	15.8676608676	-1.0323391324	-1.40725693314
95	59.7230400557	16.3662468054	16.2468027129	-0.6531972871	-0.88910033565
97	89.1167252891	16.484369595	16.6939731898	-0.2060268102	-0.934732711593
128	60.9560742693	16.3438036922	16.4357210386	-0.4642789614	-0.686210531596
129	80.0761274449	16.3289401746	16.5254908096	-0.3745091904	-0.751487912039
169	41.5987299741	16.6422209244	16.8124510757	-0.0875489243	0.658369991853
217	97.5631396311	16.4424029191	16.6513582517	-0.2486417483	0.248905742379

Table 4.1: Results of 8 brightest stars in figure 4.2a.

4.4 SNR

The signal to noise ratio of the results from this research are shown in Figure 4.4a. It is expected that bright stars have relatively high SNR since for such stars the signal dominates the noise. The slope of the trend line in the figure is exactly this way. In addition, Figure 4.4b shows a histogram of the $V^{\mathrm{PS}} - V_{\mathrm{HB}}^{\mathrm{PS}}$ magnitude. From bright to faint, the histogram first shows an increase. That increase is the main sequence, the decrease afterwards is due to the limiting magnitude of the telescope.



(a) (b)

Figure 4.4: (a) SNR vs V of the measurements from this research. A green trend line is plotted. **(b)** Histogram of V of the measurement of this research.

4.5 Kolmogorov-Smirnov Goodness-of-Fit Test

Besides the results of the measurements, it is useful to discuss the algorithm that is used. Many of the measurements involved fitting. Because the data points in the simulated data cube were without weights, the Kolmogorov-Smirnov D test can be used as an statistical approach to judge this fitting. This test is based on the maximum difference between an empirical and a hypothetical cumulative distribution. The empirical distribution $S_n(\lambda)$ is the summed spaxel and the hypothetical distribution $F_0(\lambda)$ is the result from the fitting procedure with the best fitted parameters, recall Equation 3.1:

$$\begin{aligned} I_T(\lambda) &= \sum_{i=1}^{i=3} \ell_i A_0 \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{((\lambda_0+d_i)-\lambda)^2}{2\sigma_i^2}} + (a\lambda + b). \\ &= F_0(\lambda) \end{aligned} \tag{4.1}$$

The parameters that were fitted: the overall amplitude A_0 and position λ_0, the individual line's σ_i, and the slope a and offset b of the continuum. When D is defined to be the maximum difference between the model and the dataset,

$$D_n = \max|F_0(\lambda) - S_n(\lambda)|, \tag{4.2}$$

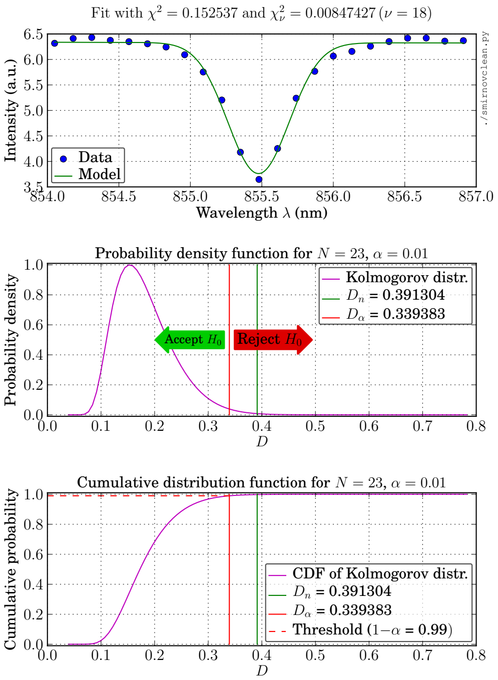
then D follows the Kolmogorov distribution. The null hypothesis can also be defined

H_0 : The spaxel data are consistent with the model with the best fit parameters;

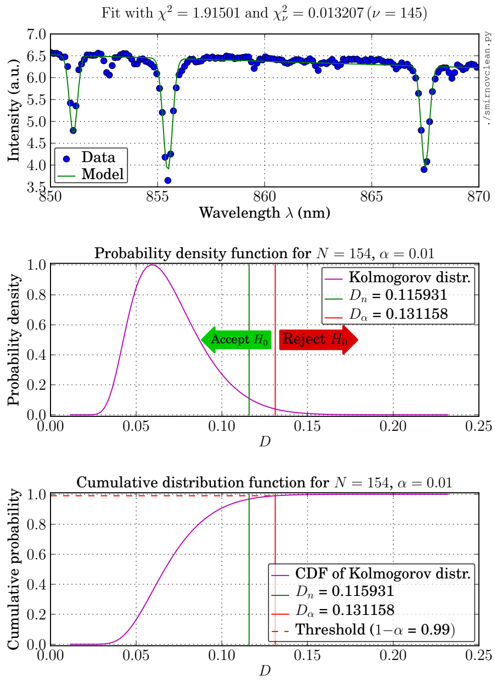
H_α : The spaxel data are *not* consistent with the model with the best fit parameters.

The Kolmogorov-Smirnov D test states that the null hypothesis is rejected at confidence level α if $D_n > D_\alpha$, with D_α the threshold value which is the solution of Equation 4.3:

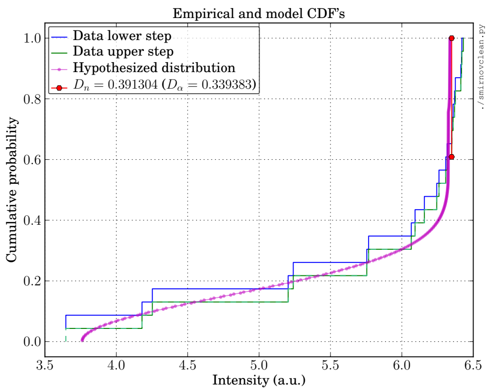
$$\text{Probability } P(D_n < D_\alpha) = 1 - \alpha. \tag{4.3}$$



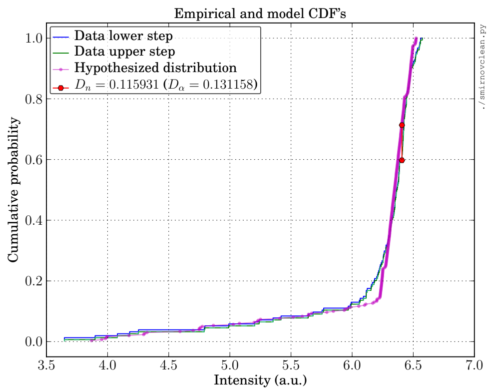
(a)



(b)



(c)



(d)

Figure 4.5: The Kolmogorov-Smirnov D test for **(a)** & **(c)** one CaT line and **(b)** & **(d)** the whole CaT for the model that is used for the fitting.

Using software from the Kapteyn Package this Kolmogorov-Smirnov D test is applied on one summed spaxel[16 and references therein]. Figure 4.5 shows the plots needed to determine D_n and D_α. The upper graph of Figure 4.5a shows the data and the best fit which is examined. The second graph shows the probability density function of the Kolmogorov distribution for the data that is in the first graph. D_α is such that the area under the probability distribution function equals $1 - \alpha$, as can be seen in the cumulative distribution function in the bottom graph. Figure 4.5c shows the empirical cumulative distribution of both the data and the model. It is used to determine the maximum difference D_n. First, the fit of one CaT line is discussed and afterwards fit of all the lines of the CaT. In the case of the best fit of one CaT line (Figures 4.5a and 4.5c), the test results

that $D_n > D_{\alpha=0.01}$ so the fit is rejected. In other words, the data is rejected to be consistent with the model with best fit parameters up to a confidence level of 99%. Inspection of Figure 4.5c shows indeed that there is quite a difference for $I \sim 6.2$ and $I \sim 6.4$. The latter yields the largest difference and results in D_n. These intensity values occur at the level of the continuum, indicating that the fit goes wrong there. The test for the whole CaT fit results in $D_n < D_{\alpha=0.01}$, so H_0 can be accepted. On the other hand, inspection of Figures 4.5b and 4.5d shows again a large difference at $I \sim 6.2$ and $I \sim 6.5$, indicating again that the fit of the continuum can be improved.
Inspecting the fit to the data a little more detailed in Figure 4.6, it is clear that the fit for 856.5 nm $\leq \lambda \leq$ 866.5 nm is too low. Moreover, the assumption that the area between the CaT lines does not contain any absorption nor emission lines may be wrong since at 852.7 nm, 859.5 nm, 862.5 nm, 863.4 nm or 866.0 nm, small absorption lines are visible. Inspection of the fit of one CaT line only (upper graph of Figure 4.5a) indicates another important detail: the transition from gaussian distributed line to the linear continuum is not fitted well. This makes the assumption that the CaT absorption lines can be represented by gaussian lines is perhaps not valid. Just for illustration, GIPSY's task `XGAUPROF` is used to show in Figure 4.7 the fit that is the result of using a gaussian profile and a Voigt profile. The data is inverted, since `XGAUPROF` only fits peaks. A Voigt profile is a convolution of Gaussian and a Lorentzian profile.

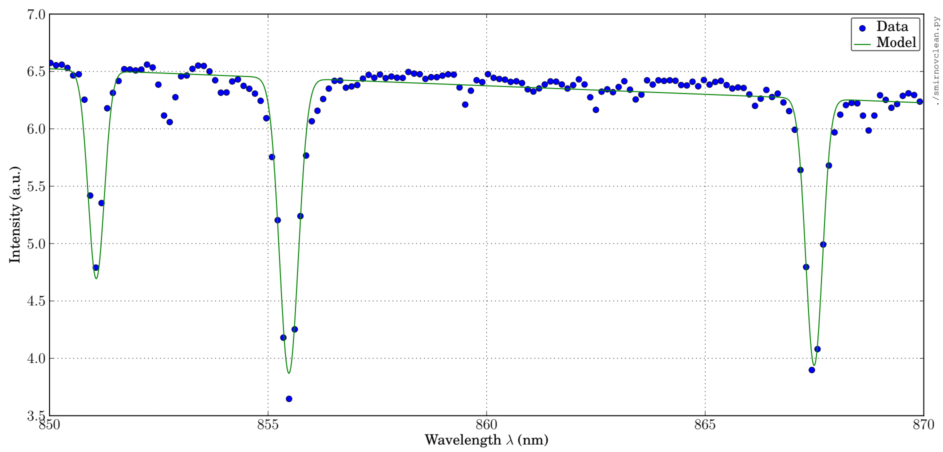
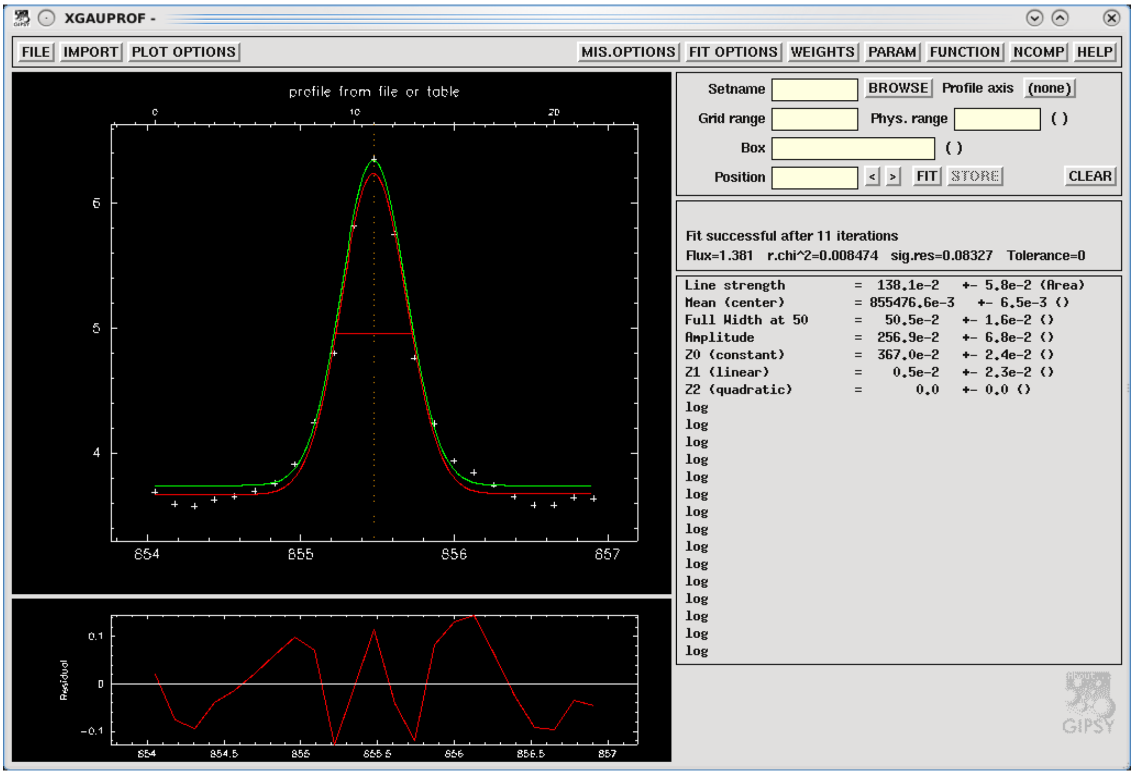
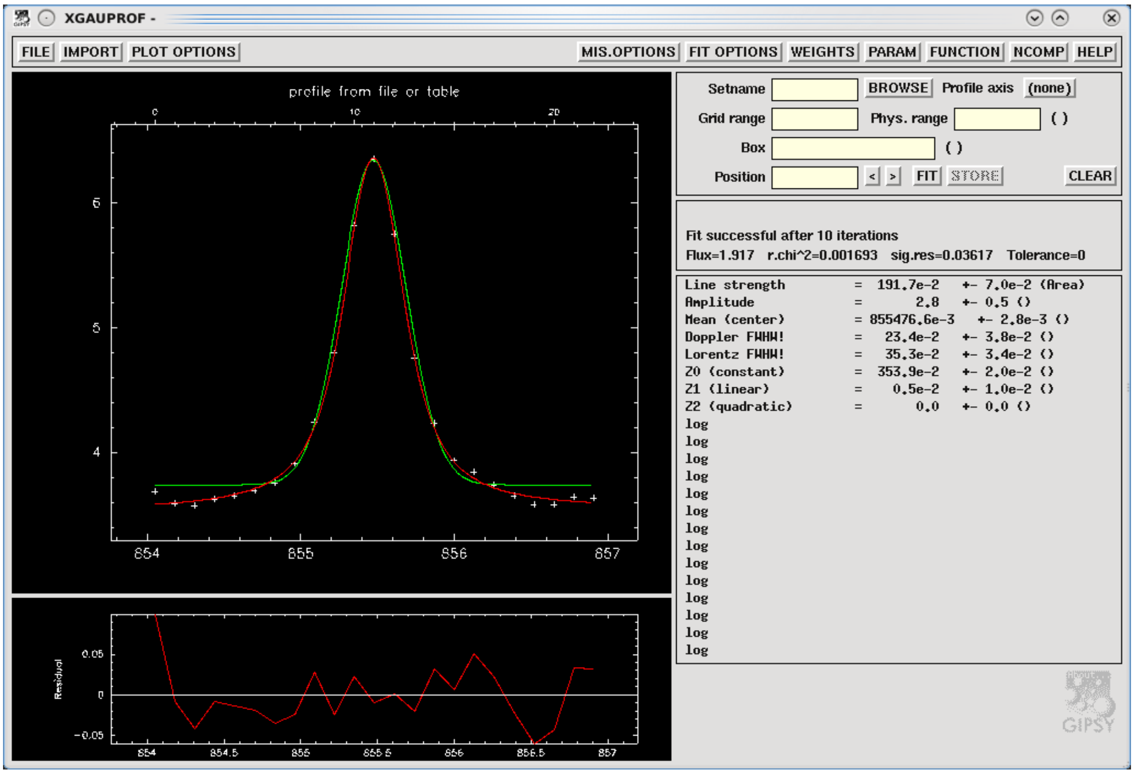


Figure 4.6: Zoom in on the data and the model for the CaT (see also upper graph in Figure 4.5b)



(a) Gaussian profile



(b) Voigt profile

Figure 4.7: `XGAUPROF` fitting using 2 different profiles as a model (the red line is the fitted function, the green line can be ignored).

Chapter 5

Summary and Future Research

5.1 Summary

The goal of this research project was to find a way to measure metallicities in a 3D MUSE data cube accurately, using GIPSY and the Ca II Triplet (CaT) lines. Multiple standard tasks of GIPSY were inspected, but none of them was capable of handling data cubes as large as a 3D MUSE data cube without preprocessing. GIPSY's binding with Python however, made it possible to use GIPSY and process these larger data cubes. Here an algorithm is described that uses both GIPSY (step 1) and its Python binding (step 2-8) is created:

1. Create a classic header that can be read by GIPSY with the task `FITSREPROJ`.
2. Read the data in the FITS data cube.
3. Localise stars in the data cube and get their central spaxel.
4. Per star, make 2 squares in x and y around the central spaxel (the squares have initial edge length $r_1 = 1$px and $r_2 = 3$px, since 1px is the lower limit and adding a spaxel on all sides of the square leads to an increase of 2 px).
   Sum the spaxels in the squares and do the following steps per summed spaxel:
   (a) Linear regression in the spectrum to find the continuum offset which is assumed to be linear in a spectral region around the CaT.
   (b) Find the CaT, using its known Gaussian line depths of individual lines ℓ_i and the fixed distances between individual lines d_i.
   (c) Fit the formula for the CaT to the spectrum:

$$I(\lambda) \;=\; \sum_{i=1}^{i=3} \ell_i A_0 \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{((\lambda_0+d_i)-\lambda)^2}{2\sigma_i^2}} + (a\lambda + b). \tag{5.1}$$

   Where $I(\lambda)$ is the intensity of the spectrum as function of wavelength λ; i runs from 1 to 3 and represents the 3 lines of the CaT; A_0 is the overall triplet intensity which can be used with ℓ_i to get line intensities; μ_0 the overall triplet position which can be used with d_i to get line positions; σ_i is the width of the Gaussian distributed line; a is the slope of the continuum and b is its offset. For the fitting, A_0, μ_0, σ_i, a and b are the free parameters. ℓ_i and d_i are known from other studies of the CaT.
   (d) Estimate the signal-to-noise ratio (SNR) of the spectrum by selecting a spectral range that is free of emission and absorption lines: $\mathrm{SNR}_{r_1} = \frac{S}{N}|_{r_1} = \frac{\mu'}{\sqrt{\mu'}}|_{r_1}$, where μ' is the mean of the intensity in the selected spectral region.

5. If $\text{SNR}_{r_1} < \text{SNR}_{r_2}$, repeat the steps of item 3 for an increased square: $r' = r + 2\text{px}$ until $\text{SNR}_{r_1} > \text{SNR}_{r_2}$ is satisfied. The summed spaxel of the final square with r_1 will be the summed spaxel of the star because there the SNR changes sign and indicates the size of the star. This summed spaxel will be used in the next steps.
6. For the stars summed spaxel, apply a linear regression to find the estimate of the continuum offset for the spectral region around the CaT.
7. Fit the formula of step 3c to the stars spectrum with a non linear least squares algorithm.
8. Measure the equivalent widths, the position of the CaT, the SNR of the star and its total flux by using the best fit values of the free parameters.
9. Save the stars measured data.

Using this algorithm, a total of 221 stars were detected in the simulated data. Inspection of the measured equivalent widths tells that they are consistent with the values that are found in the literature.

5.2 Future research

In the future, this research can be optimized. For example, the magnitude of the horizontal branch is assumed to be of zero magnitude, which of course is as estimate. As said before, GIPSY has evolved before and can do that in the future. The implemented code that is used can be optimized and perhaps implemented in GIPSY as a new task. Optimization include the usage of 2D Gaussians or other functions in the algorithm that detect the stars following the form of the spatial intensity of stars. Also, detecting stars can be improved by using the spectral information that is in the spaxels. For this, it may be useful to note that the spaxels are different when the spaxel contains flux of a star or not. This difference is shown in Figure 5.1. At last, the profile that is used to represent the CaT lines (Equation 5.1) may be reconsidered since the transition from the absorption lines to the continuum is not fitted well. Moreover, the assumption that the CaT spectral region is free of other lines is not valid since minor lines are visible.
MUSE will see first light later this year, it would be exciting to see the result of this algorithm when a real MUSE data cube will be used as input.

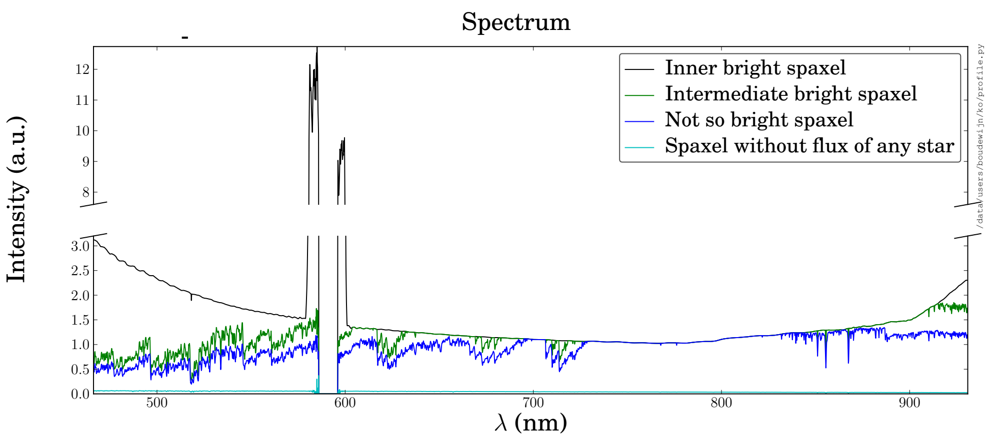


Figure 5.1: The complete spectrum from MUSE for a very bright star of one inner bright spaxel, an intermediate bright spaxels, an outer spaxel and a spaxel that does not contain flux of a star. *This figure is a copy of Figure 3.6.*

Chapter 6

Acknowledgements

It would not have been possible to do this project, without the help of some people. First of all, I would like to thank my supervisors Eline Tolstoy and Martin Vogelaar for their help and support. Eline Tolstoy gave me the insights and suggestions when it was needed and without Martin Vogelaars help, it would not have been possible to get a feeling and understanding of GIPSY and of fitting algorithms. Of course, I would like to thank the instrumental team of MUSE for the simulated data cube. Furthermore, I would like to thank other people of the Kapteyn Astronomical Institute, the computer group in particular: they provided me with enough computing power and storage, to be able to handle large data cubes.
Last but not least, I would like to thank Daniël Siepman, Sander Bus, Robin Kooistra, Mirjam Soeten and Jonathan Calluy for reviewing this report and giving hints to improve this project.

References

[1] *Integral Field Spectroscopy Wiki*, http://ifs.wikidot.com/what-is-ifs, 21 May 2009.

[2] *A Multi Unit Spectroscopic Explorer - MUSE*, http://www.eso.org/sci/facilities/develop/instruments/muse/, European Southern Observatory, 18 October 2010.

[3] *Groningen Image Processing System*, http://www.astro.rug.nl/∼gipsy/, Kapteyn Astronomical Institute, University of Groningen.

[4] Armandroff & da Costa, "*Metallicities for Old Stellar Systems from Ca II Triplet Strengths in Member Giants*", The Astronomical Journal, April 1991, vol 101, p 1329.

[5] Battaglia et al., "*Analysis and calibration of CA II triplet spectroscopy of red giant branch stars from VLT/FLAMES observations*", Monthly Notices of the Royal Astronomical Society, September 2007, vol 383, p 183.

[6] Starkenburg et al., "*The NIR Ca II triplet at low metallicity: Searching for extremely low-metallicity stars in classical dwarf galaxies*", Astronomy & Astrophysics, January 2010, vol 513, p A34.

[7] Noyola et al., "*Very large Telescope Kinematics for Omega Centauri: Further Support for a Central Black Hole*", The Astrophysical Journal Letters, August 2010, vol 719, p L60.

[8] Laurent et al., "*Design of an Integral Field Unit for MUSE, and Results from Prototyping*", Publications of the Astronomical Society of the Pacific, November 2006, vol 118, p 1546.

[9] Weilbacher et al., "*Advanced Data Reduction Techniques for MUSE*", Astronomical Society of the Pacific, 2009, vol 411, p 159.

[10] FITS Working Group, "*Definition of the Flexible Image Transport System (FITS): FITS Standard, Version 3.0*", International Astronomical Union, 10 July 2008.

[11] Bertin and Arnouts, "*SExtractor: Software for source extraction*", Astronomy & Astrophysics Supplement Series, 1 June 1996, vol 117, p 393.

[12] Stahler and Palla, "*The Formation of Stars*", WILEY - VCH, 2004, p 628.

[13] Vogelaar, "*Introduction to Programming and Computational Methods*", Kapteyn Astronomical Institute, University of Groningen. Available via the intranet of the Kapteyn Astronomical Institute.

[14] Burden and Faires, "*Numerical Analysis*", eigth edition, Brooks/COLE, 2005, p 482-490.

[15] Wall and Jenkins, "*Practical Statistics for Astronomers*", Cambridge University Press, 2008, p28-29.

[16] *Kapteyn Package v2.2b1 documentation*, `http://www.astro.rug.nl/software/kapteyn/`, Kapteyn Astrnonomical Institute, 8 March 2012.

[17] McDowell and Tody et al., "*IVOA Spectral Data Model*", International Virtual Observatory Alliance, version 1.03, 29 October 2007, p 35.

[18] Wilson and Merrill, "*Intensities of the Infrared CaII Triplet in Stellar Spectra*", Contributions from the Mount Wilson Observatory, May 1937, vol 575, p 1.

[19] Caillier et al., "*The MUSE Project from the dream towards reality*", Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, July 2010, vol 7738.

[20] Laurent et al., "*MUSE Integral Field Unit: Test results on the first out of 24*", Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, July 2010, vol 7739.

[21] *MUSE*, `http://muse.univ-lyon1.fr`, Muse Consortium, January 2012.

[22] Rutledge et al., "*Galactic Globular Cluster Metallicity Scale from the Ca II Triplet II. Rankings, Comparisons, and Puzzles*", Publications of the Astronomical Society of the Pacific, August 1997, vol 109, p 907-919.

[23] Kamann et at., "*Globular Cluster*", March 2010 (document from MUSE instrumental team).

Nederlandse Samenvatting

This is a Dutch translation of the abstract.

De MUSE (Multi Unit Spectroscopic Explorer) is een tweede generatie instrument welke opgesteld zal worden bij de VLT (Very Large Telescope) van de ESO (European Southern Observatory) te Paranal in Chili, later dit jaar. Dit nieuwe instrument combineert twee ruimtelijke coordinaten ($1\times$ 1 boogminuut) en één spectraalcoordinaat (465.00 - 930.14 nm). Dit zal leiden tot een grote hoeveelheid gedetailleerde informatie die gebruikt kan worden om astrofysische problemen op te lossen. De grote datakubussen zijn relatief nieuw voor de optische sterrenkunde, terwijl ze in de radiosterrenkunde al veel langer voorkomen. GIPSY (Groningen Image Processing System) is een goed voorbeeld van softwareontwikkeling om 3D datakubussen van radiotelescopen te analyseren. Door gebruik te maken van dit softwarepakket, in combinatie met Python en een gesimuleerde datakubus met sterren in een sterrenhoop die gemaakt is door het instrumentatieteam van het MUSE consortium, kan er tijdens dit klein onderzoek een algoritme ontwikkeld worden om zulke grote optische datakubussen te verwerken. In dit verslag wordt aangetoond hoe een spectrum van individuele sterren kan worden gemaakt en hoe daaruit de equivalente breedte bepaald kan worden. Welke vervolgens gebruikt worden om de hoeveelheid metaal in een ster te bepalen. Dit algoritme zal niet alleen nuttig zijn om de gesimuleerde datakubus te verwerken, maar ook om de datakubussen te verwerken die binnenkort van MUSE zullen komen.

Appendix A

Parameters of GIPSY's FITSREPROJ

Using GIPSY's task **fitsreproj**, the selected MUSE dataset (in **.fits** format) can be converted into a GIPSY **.fits** file with an older type of header.
A listing of the GIPSY interface, using FITSREPROJ* is provided here:

```
<USER> FITSREPROJ MAKEGDS=Y
                                                    (start fitsreproj) 23/03/11 16:57:
<USER> FITSREPROJ LEGACY=Y
<USER> FITSREPROJ FITSFILE=muse_data.fits
FITSREPROJ: Filename: muse_data.fits
No.    Name          Type        Cards   Dimensions      Format
0      PRIMARY       PrimaryHDU     48   ()              uint8
1      DATA          ImageHDU       33   (301, 301, 3578)  float32
2      STAT          ImageHDU       31   (301, 301, 3578)  float32
FITSREPROJ: File is a LOCAL FITS file.
<USER> FITSREPROJ HDUNR=1

        Axes information
==================================================
Axis 1: RA---TAN  from pixel     1 to    301
  {crpix=150 crval=0 cdelt=1 (deg)}
  {wcs type=longitude, wcs unit=deg}
Axis 2: DEC--TAN  from pixel     1 to    301
  {crpix=150 crval=0 cdelt=1 (deg)}
  {wcs type=latitude, wcs unit=deg}
Axis 3: AWAV      from pixel     1 to   3578
  {crpix=1 crval=4.65E-07 cdelt=1 (m)}
  {wcs type=spectral, wcs unit=m}

        World coordinates information
==================================================
Native sky system:                   EQUATORIAL
Native reference system:             ICRS
Native Equinox:                      2000.0
Output sky system:                   EQUATORIAL
Output reference system:             ICRS
Output Equinox:                      2000.0
Projection's epoch:                  J2000.0
Date of observation from DATE-OBS:   None
Date of observation from MJD-OBS:    None
Axis number longitude axis:          1
Axis number latitude axis:           2
Axis number spectral axis:           None
Selected spectral translation:       None

        Header Analysis
==================================================
* Header is not 'classic'. It has PC or CD elements.
* Program needs to modify: ['CD1_1', 'CD1_2', 'CD2_1', 'CD2_2', 'CDELT1', 'C
* Found (average) image rotation angle 0 (deg).
```

*mind the hidden keyword MAKEGDS=Y

```
        Re-project options
==============================================================
0. Re-project the data to the WCS of a second FITS file
1. Rotate data over a given angle (it has to create a 'classic' header first
   This re-projection removes skew.
2. Rotate image so that it is aligned to the north (it has to create a 'clas
   This re-projection removes skew.
3. Rotate to given angle. This re-projection removes skew.
   The value of FITS keyword CROTAn will be the given angle.
4. Give a number of FITS keywords and values which are inserted
   in the header and re-project the data to this modified header.
   You can select a sky definition and a projection type from a list.
5. Make classic header and copy the data (do NOT re-project).
   You lose the WCS representation of skew in the destination header.

<USER> FITSREPROJ OPTION=5
FITSREPROJ: Number of pixels in this structure is: 324170378
FITSREPROJ: Maximum number of pixels allowed by GIPSY set is 534773760
FITSREPROJ: You should be able to use this file in GIPSY!
<USER> FITSREPROJ LIMITS_RA=
<USER> FITSREPROJ LIMITS_DEC=
<USER> FITSREPROJ LIMITS_AWAV=
FITSREPROJ: Limits spatial output x: 1 to  301, y: 1 to 301
FITSREPROJ: Lower axis limits used on repeat axes: [1]
FITSREPROJ: Upper axis limits used on repeat axes: [3578]
<USER> FITSREPROJ FITSOUT=muse_reproj_data.fits
<USER> FITSREPROJ GDSNAME=
<FITSREPROJ> RFITS AUTO=Y FITSFILE=muse_reproj_data.fits; INFILES=0; OUTSET=
        e_reproj_data
                                          (start rfits)      23/03/11 17:04:
RFITS  Version 1.3  (Mar 16 2011)
File muse_reproj_data.fits into muse_reproj_data (-,-,-)
                                          (end   rfits)      23/03/11 17:06:
<STATUS>  RFITS   +++ FINISHED +++
card is too long, comment is truncated.
card is too long, comment is truncated.
card is too long, comment is truncated.
                                          (end   fitsreproj) 23/03/11 17:06:
<STATUS>  FITSREPROJ   +++ FINISHED +++
```

Doing this, the header of the new `.fits` file is:

```
SIMPLE  =                    T / conforms to FITS standard
BITPIX  =                  -32 / array data type
NAXIS   =                    3 / number of array dimensions
NAXIS1  =                  301
NAXIS2  =                  301
NAXIS3  =                 3578
EXTEND  =                    T
CDELT1  = 5.555555555555556E-05 / Appended by Kapteyn Package module Maputils 17
CDELT2  = 5.555555555555556E-05 / Appended by Kapteyn Package module Maputils 17
CDELT3  =                 0.13 / Step wavelength
GCOUNT  =                    1 / number of groups
EQUINOX =                 2000 / Standard FK5 (years)
CUNIT1  = 'deg     '           / Spatial units
CROTA2  =                  0.0 / Appended by Kapteyn Package module Maputils 17d
CUNIT3  = 'nm      '           / Wavelength units
CUNIT2  = 'deg     '           / Spatial units
CRPIX1  =                150.0 / Start x spatial coord
CRVAL2  =                  0.0 / Start y spatial coord
CRVAL3  =                465.0 / Start wavelength
CRPIX2  =                150.0 / Start y spatial coord in pixel
CRPIX3  =                  1.0 / Start wavelength in pixel
CRVAL1  =                  0.0 / Start x spatial coord
PCOUNT  =                    0 / number of parameters
CTYPE3  = 'AWAV    '
CTYPE2  = 'DEC--TAN'
CTYPE1  = 'RA---TAN'
HISTORY File modified by user 'vogelaar' with fv  on 2011-03-09T14:06:44
END
```

Appendix B

Data for technical analysis

B.1 Spatial Data

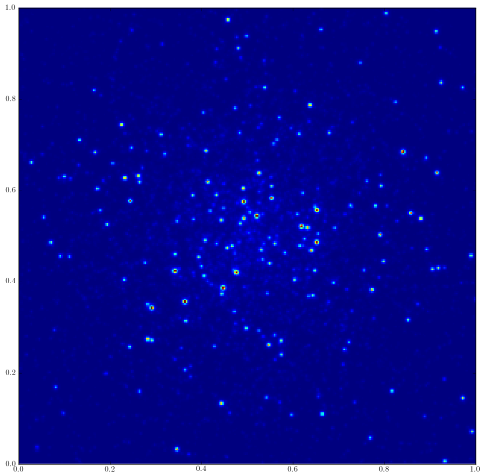
Figure B.1 shows how the finding stars algorithm works, based on the shape of a star in both RA- and DEC-direction.

B.2 Spectral Data

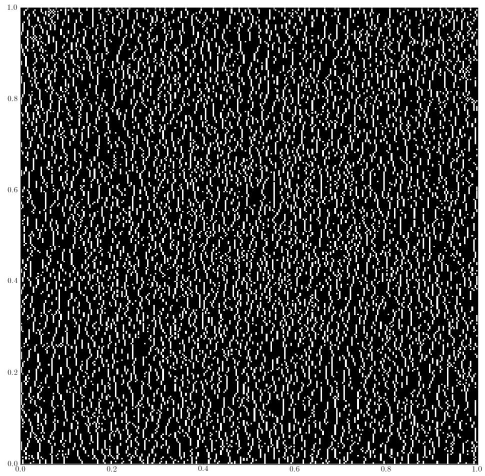
To do the spectral analysis of the spaxels in figure 3.3a, it is convenient to assign a character to each spaxels, so that it is clear which spaxel is discussed. This is done for the most particular spaxels. Figure B.2 shows the combination spaxel - character. The complete spectral range is shown Figure B.4 and a zoom on the CaT region is shown in Figure B.3.

B.3 SNR and summing Flux

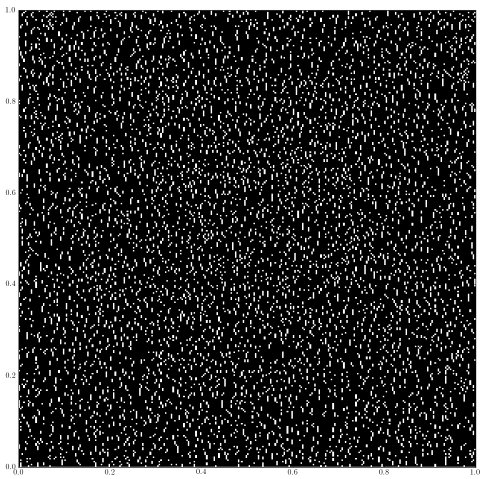
More boxes around a star and the corresponding summed spectra are plotted in figure B.6. Some of the boxes are shown in figure 3.9.



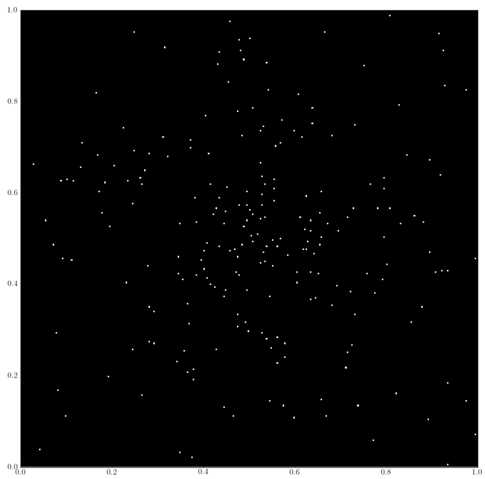
(a) Normal intensity image.



(b) Only selecting horizontally, using the derivative and the shape (within 3 px) of a star.



(c) Adding same restrictions in vertical direction.



(d) Adding clipping (at 2% of maximum intensity value).

Figure B.1: Finding stars algorithm (white dot in subfigures means that there is a star detected).

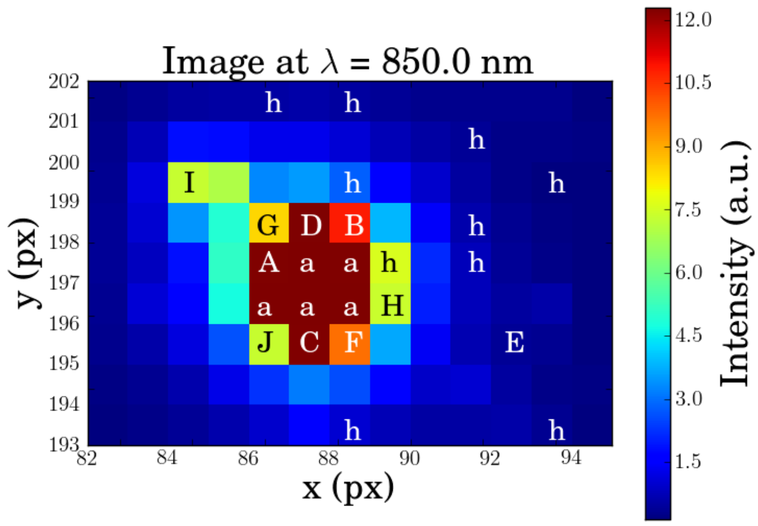
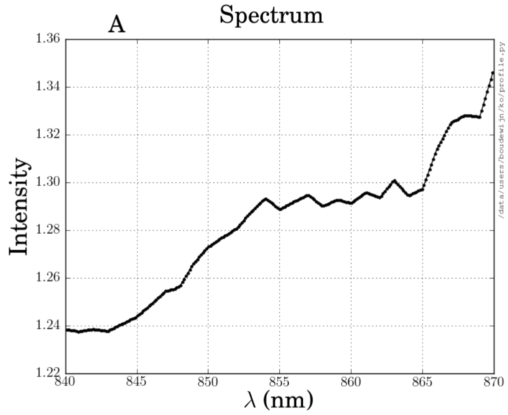
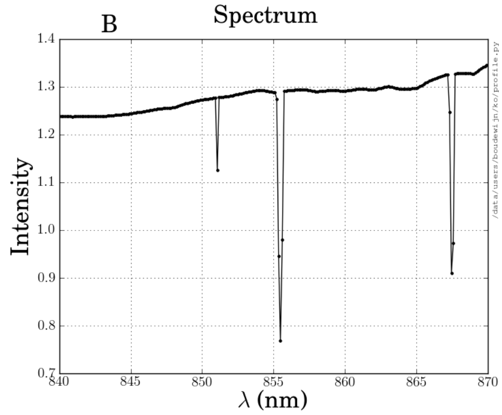


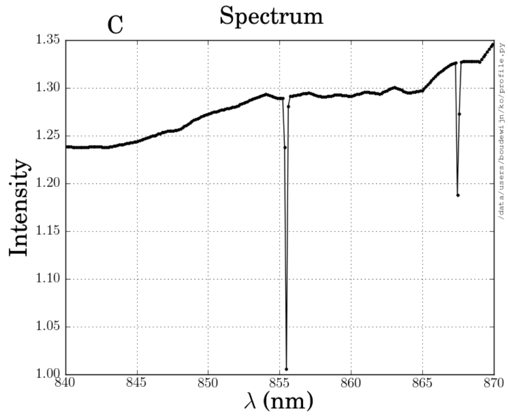
Figure B.2: Combination of spaxels and characters: this combination makes it easier to discuss the individual spaxels. Inspection of all spaxels yields a number of typical spaxels: see upper case characters (A, B, ..., H). Lower case characters are in the text represented by the upper case character, because similarity between the spaxels (that means: 'A' and 'a' are similar, spaxel 'A' is discussed in the text and showed in the appendix and represents also spaxels 'a'.)



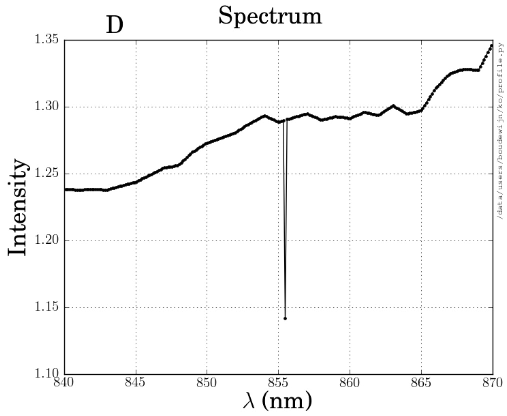
(a) Spectrum of spaxel A: no CaT visible.



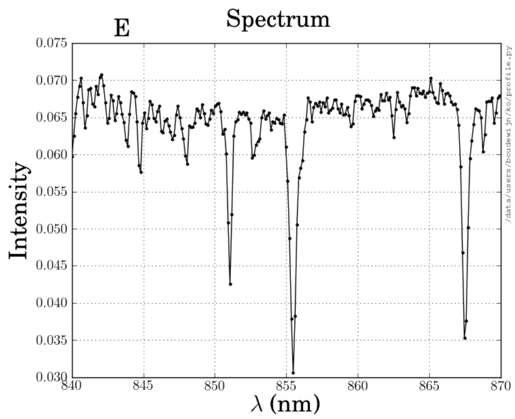
(b) Spectrum of spaxel B: high SNR and CaT visible.



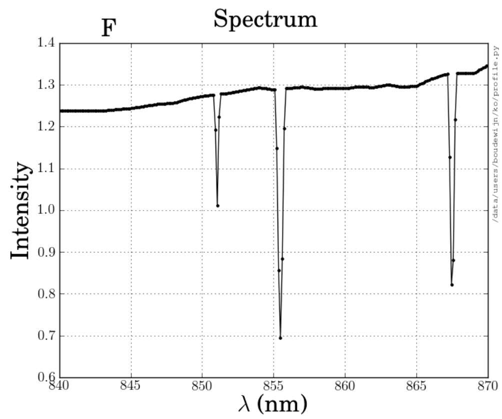
(c) Spectrum of spaxel C: high SNR and 2 lines visible (no CaT visible).



(d) Spectrum of spaxel D: high SNR and 1 line visible (no CaT visible).

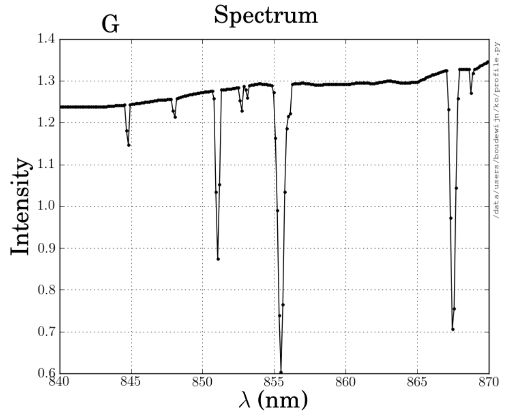


(e) Spectrum of spaxel E: CaT visible, typical SNR.

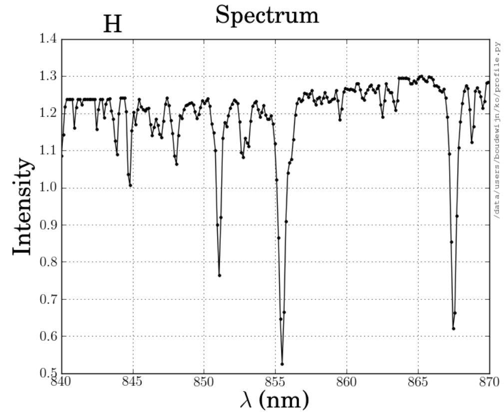


(f) Spectrum of spaxel F: high SNR and CaT visible.

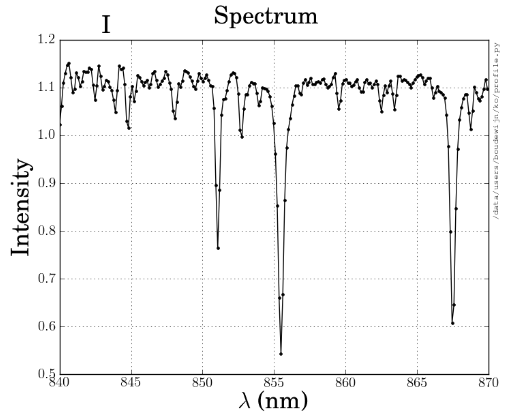
Figure B.3: Spectra of the star in Figure B.2.



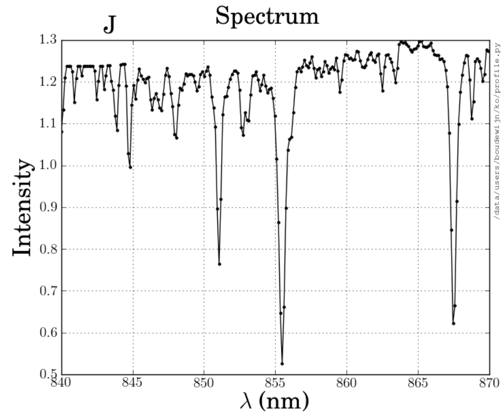
(g) Spectrum of spaxel G: high SNR and CaT and some other lines visible.



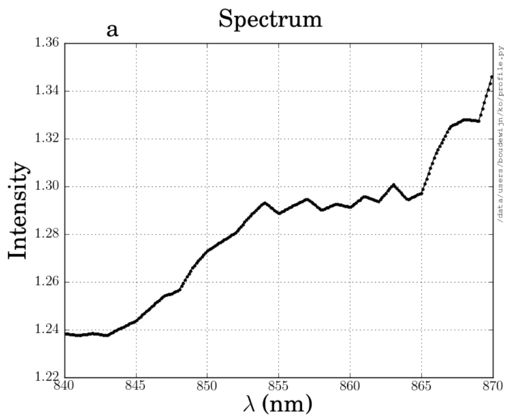
(h) Spectrum of spaxel H: typical SNR, CaT still visible.



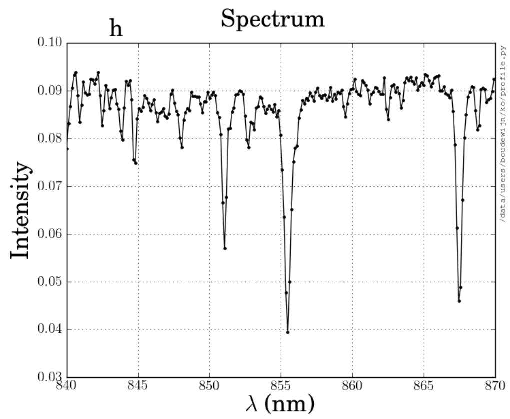
(i) Spectrum of spaxel I: typical SNR, CaT still visible.



(j) Spectrum of spaxel J: typical SNR, CaT still visible.

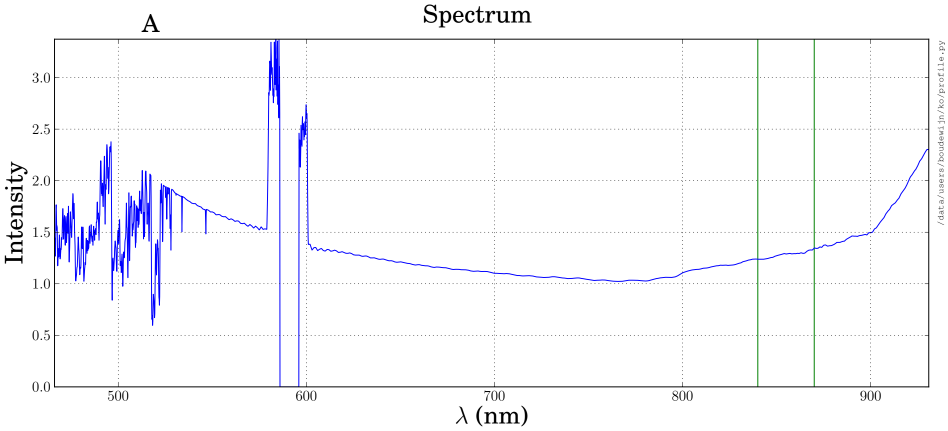


(k) Spectrum of spaxel a (the spaxel under A): similar with A.

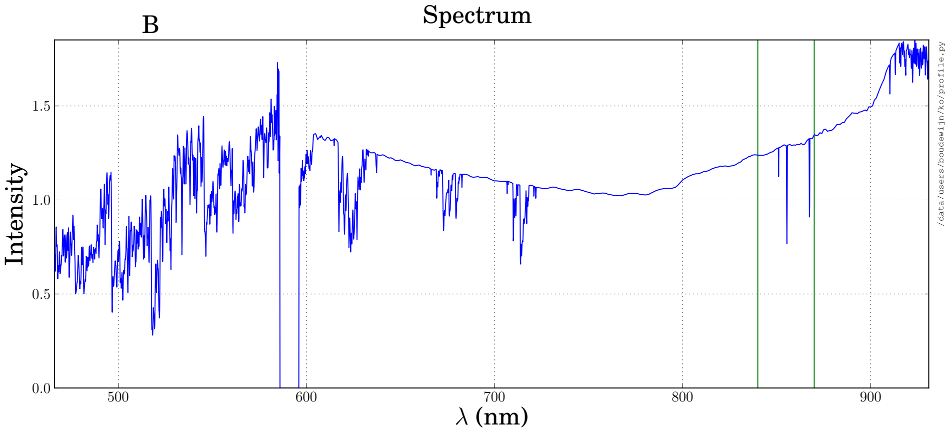


(l) Spectrum of spaxel h (upper left one): similar with H.

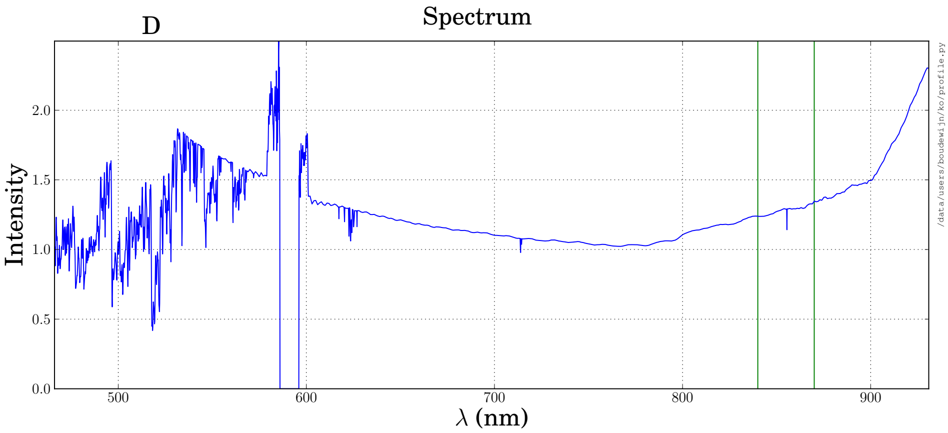
Figure B.3: Spectra of the star in Figure B.2 (cont).



(a)

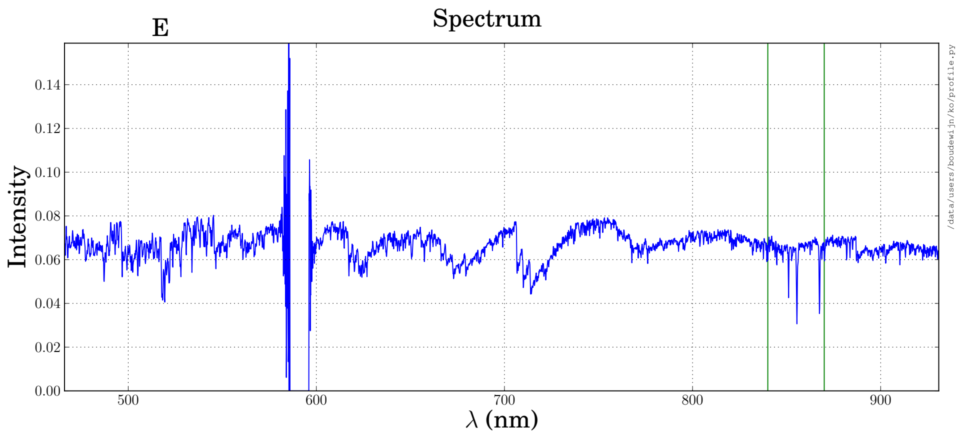


(b)

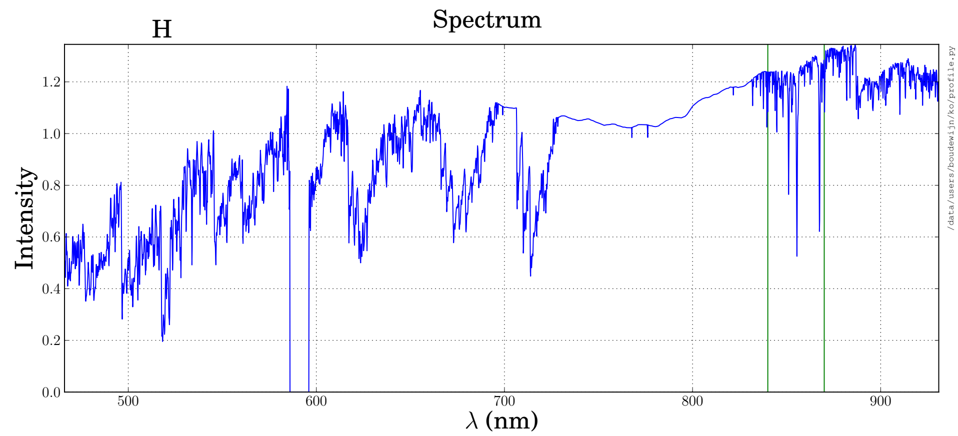


(c)

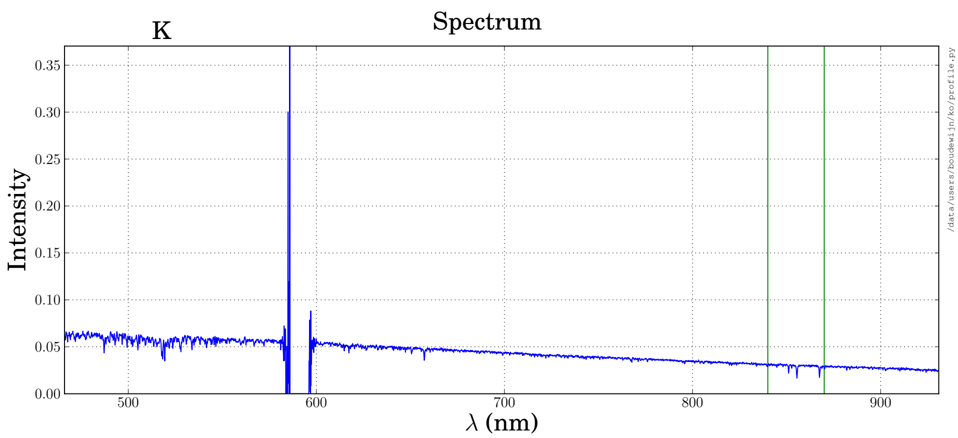
Figure B.4: The complete spectrum from MUSE for some spaxels of the star in Figure B.2. The green lines represent the spectral range shown in Figure B.3. Negative values for the intensity are not plotted since they are unphysical. The spectrum in **(f)** is added to show a spaxel that does not contain any flux of a star is therefore not assigned in Figure B.2.



(a)



(b)



(c)

Figure B.5: The complete spectrum from MUSE for some spaxels of the star in Figure B.2. The green lines represent the spectral range shown in Figure B.3. Negative values for the intensity are not plotted since they are unphysical. The spectrum in **(f)** is added to show a spaxel that does not contain any flux of a star is therefore not assigned in Figure B.2 (cont).

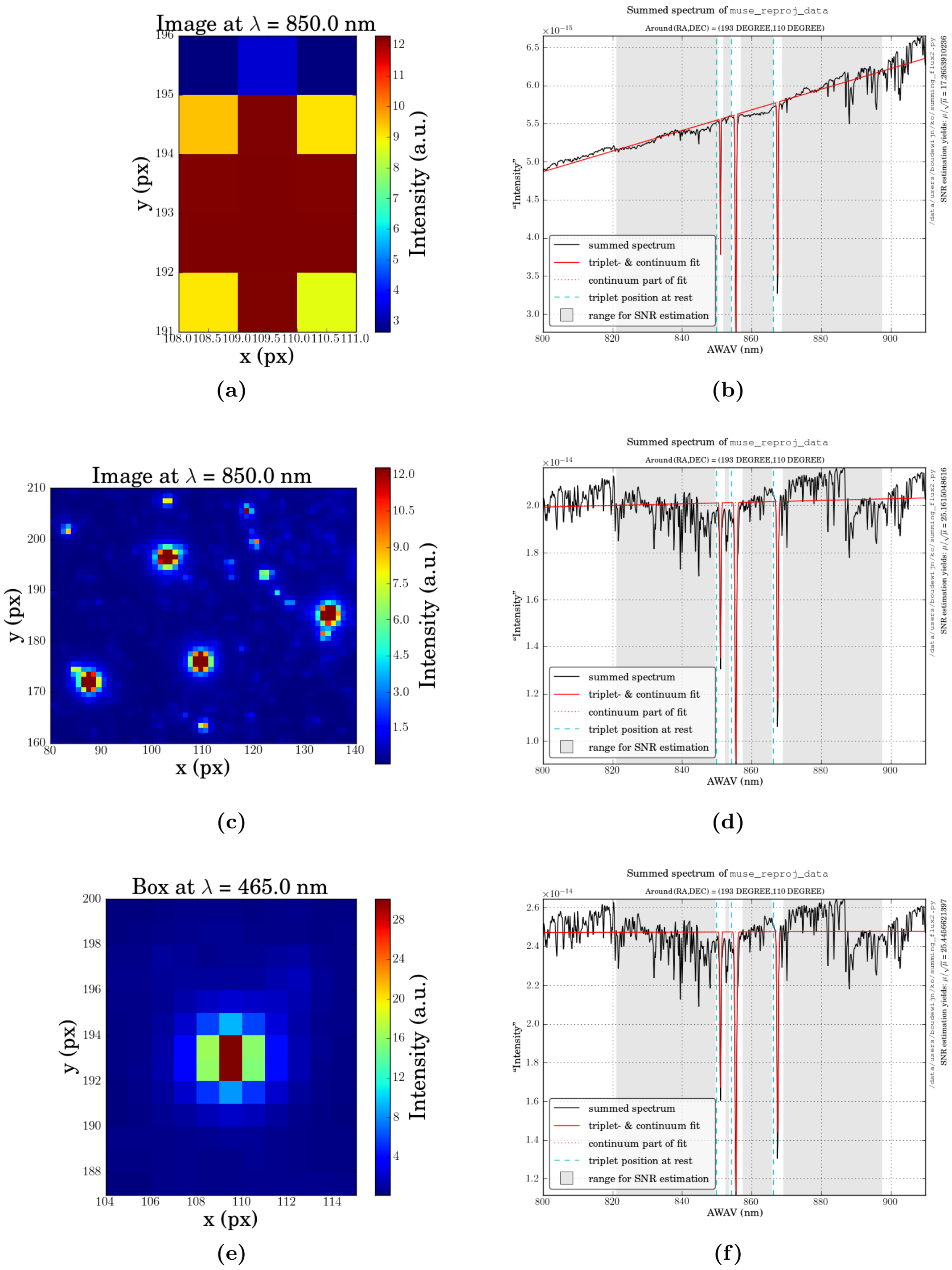


Figure B.6: Summing flux. Boxes left and corresponding summed intensities right.

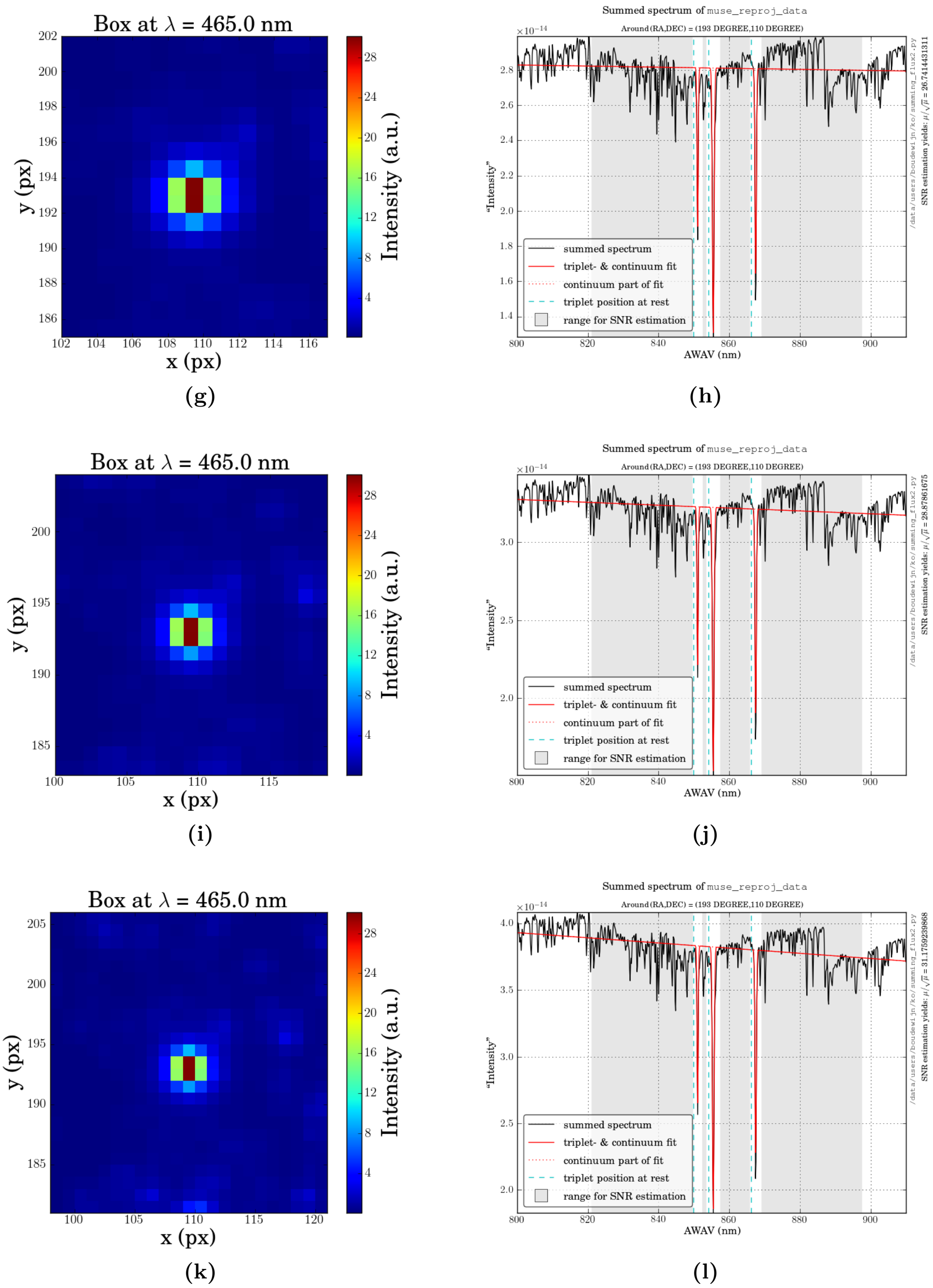
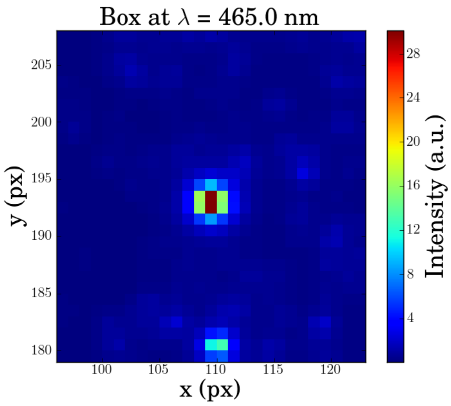
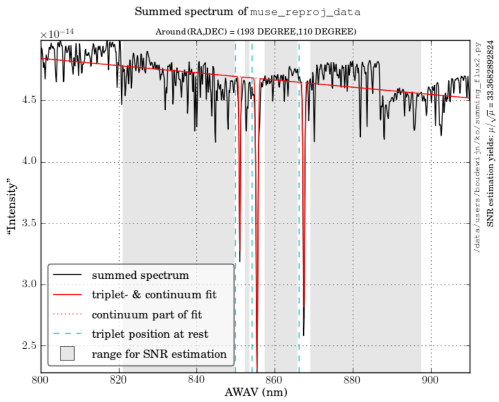


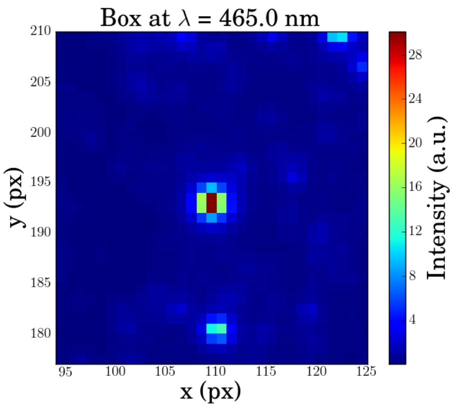
Figure B.6: Summing flux. Boxes left and corresponding summed intensities right (cont).



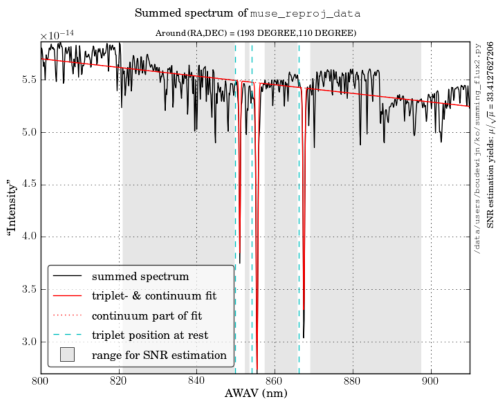
(m)



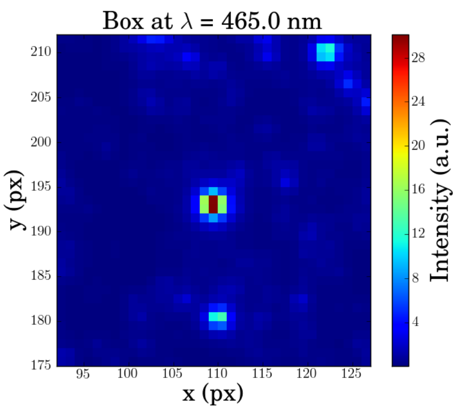
(n)



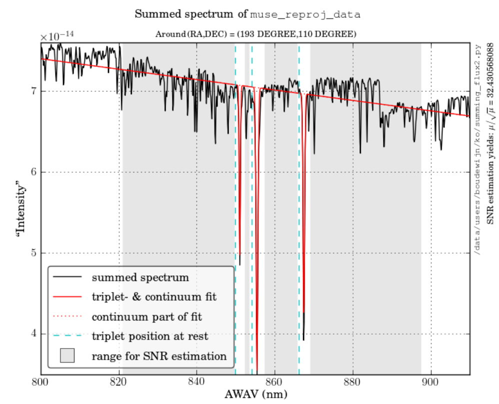
(o)



(p)



(q)



(r)

Figure B.6: Summing flux. Boxes left and corresponding summed intensities right (cont).

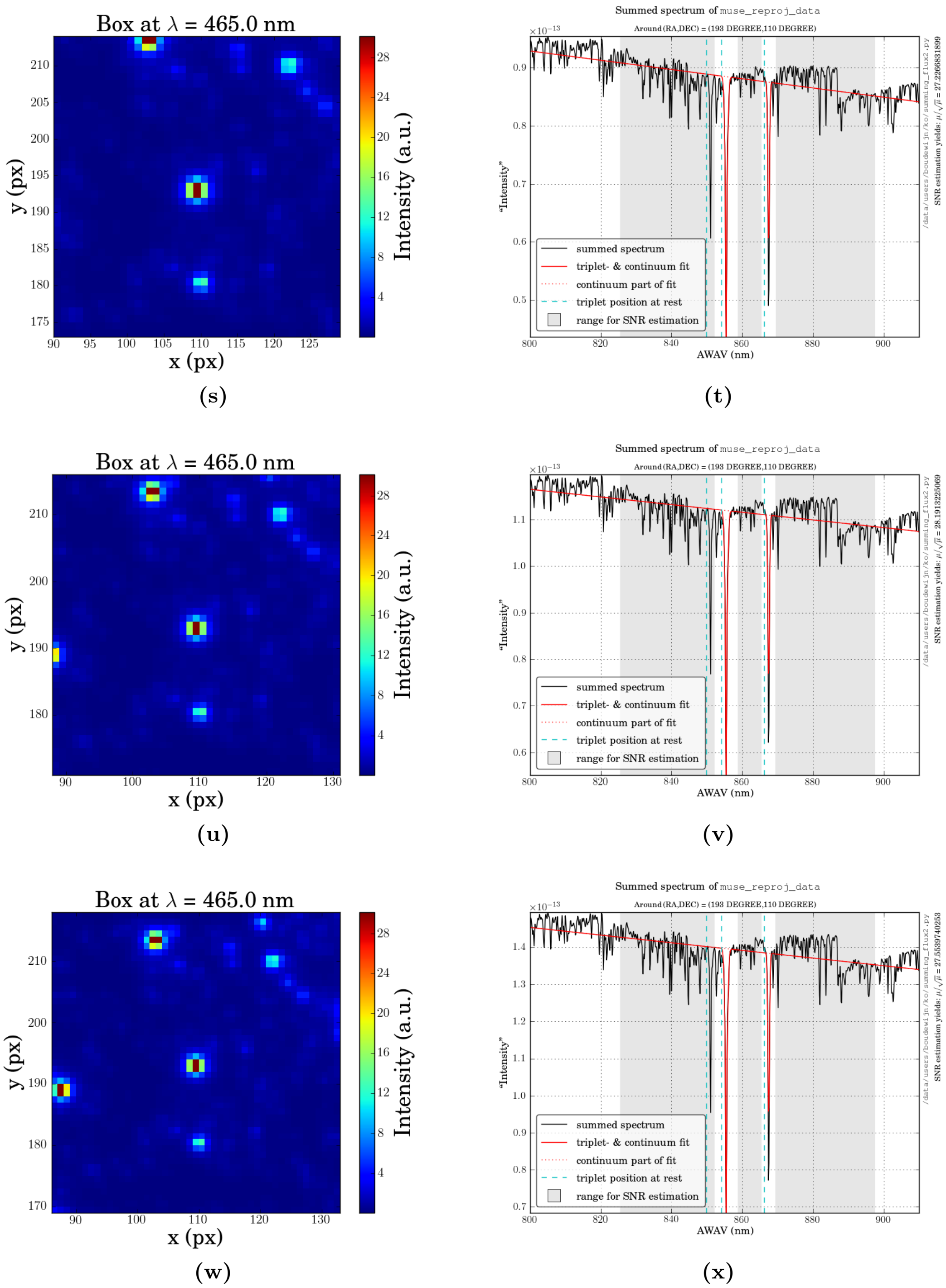
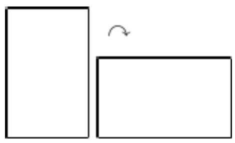


Figure B.6: Summing flux. Boxes left and corresponding summed intensities right (cont).

Appendix C

RESULTS



The rest of this appendix is printed in landscape format.

num	x	y	λ	EW_1	EW_2	EW_3	SNR	B_{PS}	V_{PS}	[Fe/H]
	px	px	nm	nm	nm	nm		mag	mag	dex
1	242	4	851.071637185	3.380807	3.36565506574	3.46291080149	52.2645073279	15.0659360012	15.375997887	1.45500998655
2	137	8	851.072027726	3.548242	3.55677176782	3.64037923061	47.3204654162	14.8971092343	15.1598719233	1.55632637291
3	74	15	851.073134522	3.521269	3.58634458801	3.65239877159	48.177350719	14.3836247858	14.5985338743	1.41655421722
4	199	15	851.06940856	3.260375	3.25242426927	3.33992955197	51.9787085747	13.7334886299	14.0261839197	0.970969073134
5	274	16	851.069294633	3.40246	3.35166573451	3.45442016989	46.1474162125	15.0964738652	15.4216404186	1.4579717398
6	150	19	851.072588186	3.811765	3.69448981211	3.81049610773	45.0900629408	14.2076337565	14.5388838191	1.51690348819
7	143	20	851.081153039	3.219419	3.19371965961	3.28426563165	49.9813615084	13.9759053294	14.2806697482	0.992310129238
8	94	25	851.072127862	3.379639	3.31658047498	3.42148096421	48.125960526	14.6588108041	14.9917267534	1.30697728699
9	144	27	851.065217646	3.468906	3.50087176488	3.58685933223	45.2783721447	14.7774395189	15.0497416042	1.47716891847
10	277	27	851.072223095	3.391084	3.41940091192	3.5078641704	47.7275083303	15.5140836249	15.8000069997	1.61783860732
11	130	28	851.069739062	3.376821	3.34320710005	3.44290843216	49.1830790741	14.2680452727	14.5857673651	1.21380292419
12	146	33	851.071268082	3.374169	3.36692766338	3.46456602803	49.2384202627	14.9065041696	15.2173512068	1.41162332405
13	161	35	851.073918393	3.764109	3.74571585568	3.81558432118	43.389167458	12.0482734271	12.2624872022	0.900648473945
14	129	36	851.063696154	3.454535	3.46329650003	3.55881165841	44.4307600879	15.076523214	15.3755153478	1.54003271165
15	225	37	851.065118302	3.323018	3.2976426578	3.39974827688	52.8977153034	13.8900731355	14.2176039107	1.07108927252
16	136	48	851.067839779	3.343564	3.32966425255	3.42480840607	54.340864775	12.6688824383	12.9761894161	0.746622909363
17	278	50	851.072063948	3.580381	3.58543601756	3.67674990977	44.5026871252	15.8741913565	16.1549155536	1.86514602792
18	162	53	851.06254813	3.453047	3.74336961166	3.8199546845	36.8059768042	15.3874948958	15.622177312	1.84762782136
19	292	53	851.07660017	3.552364	3.61273759192	3.69069111869	40.9620164245	15.5177241242	15.7631681593	1.77297678633
20	49	55	851.06806965	3.488152	3.47518593428	3.560337795	-93.6085253194	16.6955317218	16.967753529	1.99430983465
21	182	56	851.074819558	3.296025	3.21875075135	3.31974295243	51.2243546325	14.4683155723	14.798679715	1.16480543742
22	248	63	851.072802469	3.371528	3.22361123833	3.31845860288	41.6612115881	15.120547065	15.4409538416	1.34724333194
23	152	65	851.072614861	3.427165	3.41008371888	3.50192881003	48.965515236	13.6707043436	13.9638220715	1.09405780807
24	191	65	851.072087265	3.374614	3.31713690602	3.41490724497	47.6452901373	14.2528979976	14.5677897175	1.18494901087
25	142	67	851.067316667	3.493647	3.47560306233	3.56365311411	50.5812865256	13.9788588781	14.2577964465	1.23282819698
26	121	70	851.069757943	3.771705	3.77576631253	3.85541449373	36.0839568855	14.0606267482	14.302424724	1.50584235704
27	171	73	851.068801881	3.284901	3.24980466609	3.34402898885	52.9136869162	12.9837930878	13.2942031209	0.765494407031
28	191	75	851.065743076	3.600533	3.59934480676	3.6769235774	45.8723471069	13.6993194335	13.9441819215	1.24879971811
29	219	76	851.066671392	3.391512	3.14092809537	3.1928451052	85.9062273527	12.5377865959	12.7317275322	0.492674681314
30	159	77	851.070297885	3.266391	3.20908340795	3.31289382558	47.6891985778	13.9343325225	14.273043259	1.0095189645
31	67	78	851.06721916	3.382341	3.33959071638	3.43328692059	45.0338926831	14.9407618306	15.243618849	1.39322922815
32	179	80	851.069282233	3.273817	3.24987947957	3.33954069276	53.4085326679	13.5128855725	13.8113019137	0.909167494727

Table C.1: Results of this research (cont).

num	x	y	λ	EW_1	EW_2	EW_3	SNR	B_{PS}	V_{PS}	[Fe/H]
	px	px	nm	nm	nm	nm		mag	mag	dex
33	145	83	851.068536602	3.360601	3.31544975341	3.40532726657	51.1979861252	13.3880577447	13.6823540035	0.930652776177
34	204	83	851.058869326	4.339699	4.48156236733	4.49452139588	31.2060570277	13.8323178315	13.8783320456	1.97817515984
35	93	84	851.070136797	3.586474	3.66927645455	3.71906369063	50.0270743999	11.7688331831	11.9343843062	0.73215228451
36	184	84	851.059292063	3.638953	3.67225679838	3.73948365754	43.6960408547	13.912790014	14.1265535354	1.35976327617
37	111	86	851.067769374	3.439834	3.42609124427	3.5156962641	47.4951813893	13.3662551822	13.6527847377	1.01957068583
38	40	88	851.067061525	3.288036	3.29962618548	3.39664326373	48.2467119969	14.3750961407	14.689647543	1.20352330575
39	170	88	851.065781896	3.550348	3.53699607471	3.61197099409	46.2252144944	14.0614234978	14.302397177	1.29366055532
40	167	90	851.058725634	3.636796	3.66392820671	3.72555777431	40.6836421504	14.3749583635	14.5734726733	1.47582373645
41	111	91	851.070165002	3.356225	3.32305852559	3.41904896042	49.1568627089	13.414749622	13.7246694119	0.951954200234
42	74	93	851.069176617	3.397652	3.37729039759	3.46240046714	45.4541893291	14.644430238	14.9215943808	1.33194495812
43	84	95	851.070597974	3.378402	3.13505265683	3.18781312919	76.3279893536	11.312726839	11.50890122	0.143527529405
44	123	95	851.064279405	3.349755	3.31853435834	3.4019178268	52.6424531751	14.0357926559	14.3127121991	1.10801871673
45	50	96	851.066672912	3.360997	3.44631258932	3.54647841804	44.4626492275	14.5059502827	14.8173570149	1.36995577865
46	253	96	851.068580792	3.286779	3.25611007287	3.34862172121	49.4869089469	14.3703019166	14.6758915132	1.1593730395
47	96	97	851.068478625	3.356281	3.05942694292	3.09544797231	89.4690064437	11.0556793137	11.1969561478	-0.0182321860911
48	268	99	851.067283359	3.216587	3.20147187565	3.29524305395	52.9663333406	13.3132750032	13.6264730992	0.816329393761
49	157	101	851.069668897	3.498091	3.47735298827	3.56567978778	47.1034784315	12.5816355118	12.8615024493	0.841293511193
50	8	102	851.075752212	3.546724	3.61546709713	3.68587987825	37.6490587765	15.1674557193	15.3927416473	1.66774871706
51	61	103	851.067666202	3.480664	3.37977232565	3.47450852296	48.5066432123	14.174866533	14.4773893359	1.21327641038
52	39	104	851.066669997	3.391964	3.27817543774	3.37542971503	42.2040746407	14.8563835619	15.1672845761	1.31925360386
53	81	106	851.07051	3.417341	3.41211515086	3.50920802429	47.5529060215	13.2599096337	13.5667040213	0.986326049457
54	275	109	851.074761986	3.273699	3.34402522352	3.41180102035	41.8683352253	14.2507539056	14.5206059592	1.18212618541
55	158	110	851.067562584	3.391694	3.33998895582	3.43705690258	48.4307828735	13.0842687774	13.395650599	0.874675386365
56	78	111	851.070109485	3.448452	3.46990703302	3.552256539	46.0906004483	13.1390546757	13.4039591912	0.984866879936
57	238	111	851.070340458	3.626144	3.49546330129	3.54993364994	55.2337595188	11.2539073301	11.4368517102	0.441152100142
58	30	112	851.063268554	3.38028	3.38614787597	3.47236147749	46.0631773906	15.3772555184	15.6573113869	1.54740300208
59	166	112	851.067704549	3.683205	3.71845862234	3.78497459168	41.5694130411	13.6013162204	13.8111368976	1.31128676454
61	34	113	851.068782139	3.460653	3.51926175363	3.59691498129	44.9740421593	13.9338489273	14.1836194979	1.24578501398
62	70	113	851.074455312	3.544256	3.55279258446	3.63565937817	43.5014739569	15.1910683554	15.4524593111	1.63489140557
63	55	114	851.070661302	3.449827	3.44260651493	3.53149269979	47.7335519536	14.6179446464	14.901049702	1.38529925057
64	79	115	851.067892564	3.235719	3.23606475237	3.32581621883	52.8821107164	12.9257705728	13.2256768077	0.732138216371
65	124	115	851.069496035	3.023325	2.73241147226	2.76966826944	91.5024034525	10.7059002146	10.8725210884	-0.396822975146

Table C.1: Results of this research (cont).

num	x	y	λ	EW_1	EW_2	EW_3	SNR	B_{PS}	V_{PS}	[Fe/H]
	px	px	nm	nm	nm	nm		mag	mag	dex
66	160	115	851.068604317	3.440308	3.37060393885	3.47341610356	47.6641743407	12.6665382884	12.989995335	0.78991150499
67	229	115	851.067629227	3.453316	3.44932702651	3.5403850015	46.6396958374	13.8050098899	14.093608309	1.16479339215
68	135	117	851.071142895	3.407621	3.36892223402	3.467870301	50.112533397	12.7443967021	13.0584132054	0.805997874059
69	166	118	851.072391704	2.877258	2.60281337609	2.64860395128	77.4515356328	11.2425823176	11.4492447215	-0.344709062387
70	238	118	851.068218704	3.481331	3.40302046329	3.5092336018	44.346223678	14.5194983812	14.8491906607	1.3434838787
71	51	120	851.069055009	3.289663	3.25745056776	3.35489154338	50.2331607376	13.5997691644	13.9183357497	0.949393876017
72	148	120	851.066916476	3.409418	3.36751198871	3.46711260798	48.4594583032	12.5479874017	12.8638273009	0.750248590483
73	197	120	851.065028887	3.194281	3.2295590969	3.30397494981	54.4249386849	12.9438681955	13.2022028346	0.713055298789
74	158	122	851.073201956	3.502309	3.37983476777	3.47656370814	44.6652866947	13.315532249	13.6233424356	0.973708559264
75	187	123	851.066609091	3.661794	3.68596201193	3.76119190875	44.1761495476	12.4229025557	12.6570936526	0.961545297685
76	114	124	851.064669558	111.6293	1.92054452734	1.9282049421	68.5120009039	9.9452454826	10.0	-1.36999023345
77	130	124	851.06880135	3.565607	3.52123331933	3.62612884539	43.6785492698	12.6535983406	12.9719321899	0.918295457153
78	166	126	851.072134906	3.409545	3.3737847346	3.4637624121	50.2369370844	12.6969207258	12.9874041205	0.786333744888
79	73	128	851.073359769	3.359029	3.31587411011	3.41483291573	48.4120461067	13.8282714895	14.1463116569	1.06567245395
80	143	129	851.064950807	3.413457	3.32698668396	3.42447709585	46.8404295094	13.0276651042	13.3414225912	0.848148664809
81	148	129	851.066609219	3.35011	3.31383336463	3.40286618389	48.9424818855	13.1790281776	13.4712760571	0.86941913903
82	158	129	851.067201584	3.576139	3.5608026729	3.66194073895	41.9655643076	12.5844942283	12.8910924651	0.928698739387
83	128	131	851.067858225	3.492004	3.43544153928	3.54188643304	46.1684806378	12.3944350649	12.7224666785	0.77323092448
84	218	131	851.070265432	3.310805	3.28557869397	3.38158021306	50.1513746354	13.0103593618	13.3230709419	0.805886696344
85	234	131	851.068240007	3.238296	3.18261679656	3.28315024929	49.7664056186	14.040829009	14.3731960847	1.01298951764
86	242	131	851.065903802	3.507398	3.5118814255	3.60114032454	42.1686139714	14.6303447408	14.9103672995	1.44904900154
87	150	132	851.068768925	3.320136	3.29063211179	3.38300738975	50.6776267116	12.458207545	12.7610141303	0.650462959778
88	134	133	851.070118108	3.58442	3.71123394752	3.79564859679	42.4077792598	11.6055323657	11.862372813	0.764032503643
89	53	134	851.066841067	3.593688	3.66087374129	3.72722204361	39.236165023	15.1808928164	15.3929552027	1.70597833044
90	196	134	851.077183639	3.391652	3.44197366625	3.49817024664	52.4390728981	13.3531598056	13.5464520359	0.988904214988
91	126	135	851.062277364	3.329676	3.33541535657	3.41481596208	48.3325014185	13.2286232534	13.4935097765	0.890434133275
92	152	135	851.069006619	3.433926	3.39142430743	3.49081513411	47.6123786855	12.2872411681	12.6009667326	0.697177586181
93	258	136	851.064824576	3.650084	3.67934840292	3.7649274743	35.8967106671	12.8721102156	13.1340874577	1.09460041407
94	160	137	851.06949489	2.889079	2.59830469211	2.65088742392	71.3941200015	11.1553777208	11.3878983117	-0.362963304381
95	183	137	851.051792771	2.755792	2.40271686748	2.3810104516	59.7230400557	10.4985859378	10.3791418453	-0.851833635957
96	214	137	851.068964402	3.356376	3.30876446742	3.40682877272	48.1382726645	13.4976750569	13.813889097	0.965412195382
97	157	138	851.069885401	2.566248	2.17741852789	2.21640974064	89.1167252891	10.6167087275	10.8263123223	-0.8974660119

Table C.1: Results of this research (cont).

num	x	y	λ	EW_1	EW_2	EW_3	SNR	B_{PS}	V_{PS}	[Fe/H]
	px	px	nm	nm	nm	nm		mag	mag	dex
98	16	139	851.06990659	3.595254	3.64057802902	3.70193357113	47.2115465801	13.4619624964	13.6613606849	1.19830427293
99	148	139	851.069832586	3.477035	3.46603959786	3.56167357023	45.6233791268	12.9122705307	13.2114902776	0.933109456121
100	115	140	851.067641105	3.478553	3.45941295646	3.54397206319	47.3833333966	13.0241496304	13.2952985209	0.946005472142
101	264	140	851.068215317	3.47528	3.54610629356	3.61825040105	42.5549772632	12.5612541416	12.795208025	0.876007525456
102	104	141	851.070414865	3.390971	3.37506408991	3.46776296614	52.125061417	13.3527595889	13.6507128927	0.975444655252
103	133	141	851.06427631	3.447839	3.48585009224	3.55640824271	46.8909350666	13.1431464408	13.3755077365	0.98569664598
104	249	141	851.055176411	3.399092	3.38085996973	3.4760507906	49.8292766794	14.1856695946	14.4892063392	1.21776123967
105	58	143	851.068772741	3.44147	3.43156309147	3.52234835293	45.5685844396	14.3349378685	14.6241564876	1.29844350245
106	146	143	851.064874929	3.39791	3.44227771123	3.49815693005	51.4933151534	13.0930423051	13.2850161783	0.915411797985
107	186	145	851.066881008	3.703166	3.75212918983	3.81545840648	43.4278677468	12.4050491719	12.604860341	0.999827214415
108	190	146	851.07078317	3.531316	3.53190486745	3.61470816733	46.2139230325	12.8550661139	13.1173372359	0.958911900926
109	208	146	851.070767346	3.634002	3.471016422	3.55042872722	59.3700079514	11.7323831687	11.9845365393	0.584841355133
110	151	149	851.069953957	3.445106	3.43789208778	3.51741820456	49.5663129976	11.5635629924	11.8227840755	0.510192524296
111	197	150	851.073156633	3.269317	3.23544840967	3.32336727821	52.8582519735	13.4003152883	13.6951753654	0.863000285561
112	238	150	851.08132032	3.415595	3.38182400009	3.47278589746	47.6484533525	14.4470647858	14.7397762207	1.28730933866
113	170	151	851.068160443	3.318218	3.34958262909	3.40184978478	55.9592590023	12.7812572248	12.9665971875	0.742584030101
114	165	152	851.06773976	3.34113	3.29174154004	3.38305266287	50.823619114	12.5848559775	12.8846569634	0.68578885017
115	152	153	851.070010156	3.400634	3.45051499747	3.53084945379	50.6108996038	11.5785869077	11.8392144041	0.526283134763
116	188	153	851.065157238	3.34713	3.31151154822	3.40524047824	50.9924020034	12.6972196377	13.0019681303	0.737285117147
117	122	154	851.060374046	4.043631	4.13517953349	4.1850752455	33.9786894385	12.5453906342	12.6966392924	1.3568457275
118	21	155	851.064477991	3.457874	3.41772059949	3.51770815304	48.3220738468	14.4486735022	14.7621731504	1.32917661027
119	145	155	851.068077098	3.514618	3.51095501903	3.59882591816	47.6865441425	11.9274575921	12.20408911	0.685535105747
120	196	155	851.067306641	3.411833	3.38039932487	3.47943441436	46.8390618309	13.1372239425	13.4507374733	0.926614517756
121	130	156	851.069139976	3.784668	3.85519466713	3.92216035067	40.8226963335	11.8079936332	12.0132634457	0.925531194154
122	161	156	851.066734762	3.466283	3.40135290534	3.50637269667	46.7087684486	12.4466511158	12.7735830173	0.757000242569
123	168	156	851.070447985	3.415892	3.39787987909	3.48601069739	49.094616266	12.1174298338	12.4017798575	0.64181306153
124	140	158	851.070352728	3.789645	3.89695946846	3.97008775649	39.1607556646	11.9197066452	12.1392604652	1.00047652598
125	185	158	851.068406312	3.465238	3.45337003235	3.54233429451	46.2670968483	13.0374173318	13.3203243591	0.949673243353
126	187	158	851.068536525	3.395773	3.35964052023	3.45509759792	48.1401611906	12.8588002455	13.164608243	0.826198453222
127	120	159	851.069771717	3.53009	3.58281158414	3.63974093331	51.2573324856	11.4873430327	11.6765396938	0.586596685447
128	137	159	851.06242146	2.879378	2.5532108356	2.57072194604	60.9560742693	10.4761428247	10.5680601711	-0.648943831903
129	159	160	851.07147618	2.728848	2.43885026307	2.47927218233	80.0761274449	10.461279307	10.657829942	-0.714221212346

Table C.1: Results of this research (cont).

num	x	y	λ	EW_1	EW_2	EW_3	SNR	B_{PS}	V_{PS}	[Fe/H]
	px	px	nm	nm	nm	nm		mag	mag	dex
130	268	160	851.071073393	3.437129	3.41026500882	3.5049249754	47.7592338505	13.7534152596	14.0536363648	1.12074759338
131	192	161	851.063107167	3.395995	3.36222293409	3.45837321182	44.7582052399	13.2302326391	13.538031587	0.933931999094
132	175	162	851.06953735	3.402034	3.376219021	3.47122666751	46.3639709118	12.9901072725	13.2935749711	0.876906814816
133	103	163	851.069006952	3.467285	3.43558231653	3.52695635006	43.167456897	13.5700497313	13.8604970105	1.08719297145
134	142	163	851.066640835	3.551545	3.52918701029	3.62259074556	44.0459425404	12.7777519401	13.0673096726	0.947096616386
135	27	164	851.068271556	3.294197	3.30165072394	3.37537168002	45.4789170177	14.9825725815	15.2336396661	1.34824278771
136	298	164	851.069571372	3.338147	3.30560655319	3.40271776087	49.1121425594	14.549883652	14.8640158282	1.2579295554
137	33	165	851.071330924	3.275101	3.21532058397	3.32161401175	47.7611989129	13.416085478	13.760573542	0.87178873155
138	118	165	851.071717712	3.932118	2.11270606004	2.13653766287	105.652086344	10.9601941549	11.1015365084	-0.883580081167
139	160	166	851.068995143	3.574578	3.4708157762	3.57484328383	46.5261013965	12.3246798339	12.6440754631	0.781221636811
140	157	167	851.072266167	3.828127	2.56009460844	2.61238724562	85.557811734	11.2472341903	11.4811294284	-0.370461937186
141	240	168	851.069554873	3.312086	3.24460912745	3.34741778136	50.7549650884	13.7611227037	14.0942320302	0.989987579578
142	83	169	851.069059071	3.461102	3.47183801231	3.56066296313	45.9784352828	12.5813141731	12.8630075069	0.837083343133
143	165	169	851.066156402	3.573603	3.5461083684	3.63868736461	45.2987250749	13.4014040449	13.6875018671	1.1362706483
144	120	171	851.070279952	3.483829	3.44952612195	3.54853982888	44.8938703847	13.2212478394	13.5299734716	1.00974954798
145	276	172	851.061122335	3.103703	3.05870821984	3.16067796408	-153.124015548	14.7011620152	15.0477325882	1.09453141775
146	280	172	851.070189377	3.412489	3.44531706966	3.51908100053	49.1072728015	14.1504811595	14.393594653	1.23813140516
147	141	173	851.068381255	3.458418	3.46629751107	3.55027999879	47.1044059249	12.2821337811	12.5514594306	0.742345079988
148	181	173	851.069146065	3.347228	3.38686487234	3.47591232058	47.3883329389	12.622121381	12.9095503278	0.7755113372
149	190	173	851.062839196	3.402422	3.37567117579	3.47242164295	45.5544886843	13.4891346988	13.7972929284	1.01903852887
150	272	173	851.070853423	3.218655	3.13749901964	3.24100206167	50.1694292065	14.6437901074	14.9874788983	1.14757453354
151	103	174	851.066431893	3.405082	3.39182525714	3.48517177248	48.2357310796	13.0116758999	13.3100849317	0.894558609793
152	195	174	851.066941766	3.426797	3.43060350055	3.52107268417	46.1813198772	13.4795620471	13.7679831716	1.05636158241
153	227	174	851.07026437	3.331984	2.9194589429	2.95358752959	118.747153464	11.3090593273	11.4465921531	-0.0719392018013
154	115	175	851.06930943	3.391373	3.39439981166	3.47122121788	52.3048551627	13.0325846722	13.2866934189	0.882966119765
155	143	175	851.070827857	3.462887	3.48647846457	3.57588636441	45.8672416402	13.0908609375	13.3727758217	0.99377419615
156	122	177	851.068004255	3.377613	3.36623422929	3.45337748133	49.1608057475	12.7292630247	13.0130504751	0.785664166448
157	237	178	851.06649138	3.337199	3.29129681601	3.3917318743	50.9175248023	13.5216854772	13.84516589	0.959891338366
158	69	180	851.074061899	3.518978	3.54666938174	3.617648564	41.6521640622	14.8718652262	15.102142411	1.52562319905
159	181	180	851.07074037	2.938063	2.65823490255	2.7025474029	80.4432123221	11.3603338083	11.5580045676	-0.265961699374
160	124	181	851.067106492	3.409794	3.37370518681	3.46671660243	48.6371958162	12.9325777878	13.2311995792	0.856251388771
161	207	182	851.068563974	3.268794	3.20796973337	3.30908493066	50.2666254143	13.3310930006	13.6627691608	0.835499847868

Table C.1: Results of this research (cont).

num	x	y	λ	EW_1	EW_2	EW_3	SNR	B_{PS}	V_{PS}	[Fe/H]
	px	px	nm	nm	nm	nm		mag	mag	dex
162	127	183	851.067882949	3.486269	3.40656152032	3.48036681773	51.535530135	12.2751859181	12.5183253437	0.675968885526
163	134	185	851.066231897	3.314518	3.10927842263	3.15181160308	69.4969710914	11.4177037683	11.5835028008	0.137354000013
164	148	185	851.059000469	4.052508	4.1037685052	4.19070773895	34.2248445253	13.1519097279	13.3949132146	1.54213710865
165	232	187	851.06798788	3.315571	3.41767057356	3.50591337291	43.2092943489	14.4127473201	14.6965123598	1.30547481696
166	133	189	851.066856625	3.614723	3.61236404401	3.70487477025	42.6571875636	12.3728616792	12.6550993265	0.903821048626
167	163	189	851.067611375	3.430342	3.46409002832	3.53578006031	45.5936110339	13.4300177491	13.6661224666	1.04888292559
168	193	190	851.064473773	3.571472	3.53338244503	3.63413434999	45.1888374435	13.0603328984	13.3677400371	1.03862298427
169	190	191	851.069979652	3.519347	3.9420173636	3.99667302034	41.5987299741	10.7745600568	10.9447902081	0.695636691546
170	109	194	851.067019185	3.373183	3.36986927391	3.45303693327	49.0748002057	13.8847580977	14.1577276221	1.10945482953
171	204	195	851.071494743	3.30512	3.26971910196	3.35878773599	49.2452115063	13.7889060782	14.0843886833	1.00326686192
172	84	196	851.070026164	3.265736	3.27204786396	3.35054303855	54.0542030302	13.5826260239	13.849289315	0.934459868222
173	263	196	851.072250746	3.858337	3.81902178491	3.9125469585	45.5804662018	12.6765700133	12.9431105855	1.16723018797
174	87	199	851.063585538	3.481427	3.55681869318	3.63620826905	40.3167986542	14.6927182343	14.9442986655	1.49380636758
175	142	201	851.067243229	3.474498	3.4765274298	3.5722902576	45.357354254	12.5283621175	12.8274620278	0.834253089492
176	219	201	851.054140848	3.332137	3.27658748781	3.35795983361	43.4764617972	14.3772160845	14.6526248535	1.16593998018
177	147	206	851.070691503	3.665657	3.67099466978	3.73963564098	41.902542794	11.7709787281	11.984767332	0.756147817431
178	256	206	851.065480791	3.436663	3.41098285011	3.49895524728	46.0019542873	15.1055089094	15.3886687381	1.49438187951
179	110	207	851.06770415	3.326136	3.30888834275	3.41366738121	45.9129974566	13.8468753251	14.1797738656	1.07150883911
180	142	209	851.07168528	3.497459	3.46148502901	3.56381766368	45.0235495983	12.6702882329	12.9864903116	0.868688856519
181	149	212	851.071133223	3.400638	3.36588453593	3.46091816738	49.0963639147	13.3419979275	13.6463652729	0.96716965031
182	23	213	851.068615143	3.397604	3.38510695921	3.49011227404	47.8556583854	14.2388835019	14.5673428508	1.24782020941
183	158	213	851.072653237	3.258836	3.19590851408	3.29898002611	47.146311099	13.7960114434	14.1340645716	0.958463541041
184	168	216	851.069065211	3.419413	3.41589880201	3.50653188861	48.8217116279	13.2306538954	13.5203898318	0.973771280504
185	161	217	851.068624933	3.326423	3.2924773104	3.39035329347	49.0101667832	13.6189274663	13.935943878	0.985367261738
186	84	219	851.067837276	3.372296	3.33417609956	3.43310200801	51.6542985776	13.5745247674	13.8909707111	1.00985971957
187	87	220	851.065552665	3.388423	3.35410361463	3.45029191778	51.4694050862	13.4913158998	13.7993595206	1.00039367526
188	173	220	851.072157889	3.262858	3.21466125778	3.30951812832	52.5184303432	13.5294607407	13.8444769096	0.889803627617
189	217	221	851.070265533	3.428412	3.39305727483	3.48926494579	49.5420489047	13.8919820429	14.1974674999	1.14678862505
190	164	223	851.069245439	3.512152	3.47862628933	3.56799679768	43.7913191335	14.5726224241	14.8551036736	1.40427135277
191	73	224	851.066544414	3.407533	3.39982055999	3.49109125146	46.8750872372	13.8573516011	14.1498136676	1.13714872582
192	128	224	851.067396927	3.591594	3.53385520184	3.63374998119	44.5833853746	14.1005427548	14.4057916654	1.33097721352
193	107	225	851.064486312	3.439433	3.43418560435	3.52833316483	50.624702715	13.426387371	13.7237088275	1.04866466426

Table C.1: Results of this research (cont).

num	x	y	λ	EW_1	EW_2	EW_3	SNR	B_{PS}	V_{PS}	[Fe/H]
	px	px	nm	nm	nm	nm		mag	mag	dex
194	214	226	851.066537561	3.320156	3.30087932803	3.39740553488	47.4782565468	14.0455218793	14.3583916582	1.11112843062
195	173	229	851.069375885	3.652131	3.82448993337	3.89128061272	41.6437684624	11.5547559262	11.7606462931	0.827297036424
196	168	233	851.071461696	3.478055	3.4336159009	3.53275126033	48.4490951818	13.5624125778	13.8724579288	1.09224570369
197	213	236	851.070355593	3.371318	3.349791281	3.44669298418	47.8012672359	14.2060195105	14.5164003365	1.19883141144
198	113	237	851.074888289	3.664841	3.5833529106	3.69372841389	42.0224665764	13.681218506	14.0077961796	1.26707118695
199	109	239	851.066372084	3.490186	3.60103788421	3.64221742986	43.6639291716	11.6024058162	11.7455885939	0.61515008624
200	57	242	851.068000254	3.37419	3.35016396938	3.44452885982	50.7274150635	14.0346310536	14.3381826484	1.14785707864
201	113	244	851.070647438	3.614062	3.54664951662	3.64895396285	45.6793394557	14.364887266	14.6749046268	1.41907867387
202	280	246	851.064351828	3.39311	3.37205444755	3.46044639615	48.5962470326	15.1274935092	15.4141054648	1.46747247011
203	24	251	851.070528801	3.308077	3.27629044954	3.37282669313	52.7920154255	14.542056257	14.8566887106	1.22981508368
204	246	253	851.079020906	3.120768	3.0786736062	3.17244027841	52.1890618442	13.8763306121	14.1990711916	0.86950855678
205	79	254	851.061987565	4.252644	3.88650304754	3.93681297385	39.7408863038	15.0076161949	15.167220441	1.8339083256
206	197	257	851.06768833	3.384248	3.33491370407	3.43299957424	50.6885928654	13.6267266359	13.9411336994	1.02426509222
207	163	258	851.067782527	3.491396	3.4623136956	3.55196418797	44.7477118707	14.7939745592	15.0780168394	1.45281181074
208	292	258	851.066366675	3.267109	3.22803217725	3.32593194161	52.6409568728	14.5764758481	14.8982699916	1.19965704193
209	172	261	851.066445725	3.281548	3.2030517237	3.31115258344	48.8724653242	14.4049582146	14.7546542771	1.14172053956
210	221	261	851.094312376	3.401342	3.6407119437	3.77146980228	-44.871267552	18.1451990355	18.5120368409	2.59490954263
211	133	262	851.064652508	3.714863	3.73861656434	3.82035418334	35.8174551124	12.9339937631	13.182935497	1.15882176494
212	29	268	851.072437564	3.352124	3.34655214272	3.43803553614	49.3932533433	15.1983458895	15.4949857994	1.4691665798
213	139	268	851.065154193	3.864301	3.9049104115	3.98760435589	30.4339755412	13.1400893756	13.3835068013	1.36206201291
214	200	268	851.070445486	3.37808	3.38594895866	3.46447900287	53.0328112734	13.7062434412	13.9657644031	1.06750755898
215	179	269	851.069541585	3.534326	3.47481794291	3.57802600836	46.476633474	14.4044321795	14.7215253433	1.36939287524
216	267	270	851.069536842	3.461168	3.58106284218	3.62829929328	44.2089282446	12.9051186402	13.0670123511	0.972350017662
217	298	280	851.069881536	3.740228	3.52329551786	3.58789370648	97.5631396311	10.5747420515	10.7836973841	0.286172442072
218	231	284	851.063634553	3.107164	3.05990176725	3.14882565137	51.8197760324	14.4439638176	14.7549970065	1.00740722123
219	12	290	851.065744967	3.4482	3.53426611184	3.59838862849	27.8806413415	16.4469470454	16.6599105421	1.95035889441
220	104	292	851.079009924	3.433297	3.24489970574	3.36830230093	-105.966299895	15.7989883992	16.1831428507	1.58754190969
221	112	295	851.074935617	3.430277	3.4311356721	3.52028456216	44.6048095537	15.357507646	15.6424763887	1.58410625413
222	280	300	851.072553419	2.74085	2.40622964933	2.44711262326	81.8698558793	11.4345196276	11.6354284322	-0.467432753552

Table C.1: Results of this research (cont).

Appendix D

Python script

In order to provide the possibility to do this research once more, the python code is provided here. It should be executed from within GIPSY:

```
<USER> ko.py
                                              (start ko)     24/02/12 08:30:38
<USER> KO.PY SET=muse_reproj_data
<USER> KO.PY SPECTRA_OUTPUT=y
SPECTRA_OUTPUT = y
<USER> KO.PY MINX=620
<USER> KO.PY MAXX=910
MINX = 620
MAXX = 910
<USER> KO.PY SNRMINX=820
<USER> KO.PY SNRMAXX=845
SNRMINX = 620
SNRMAXX = 910
<USER> KO.PY TRIPLET_MU1=
<USER> KO.PY TRIPLET_MU2=
<USER> KO.PY TRIPLET_MU3=
<USER> KO.PY TRIPLET_LD1=
<USER> KO.PY TRIPLET_LD2=
<USER> KO.PY TRIPLET_LD3=
TRIPLET_MU1 = 849.8
TRIPLET_MU2 = 854.2
TRIPLET_MU3 = 866.2
TRIPLET_LD1 = 3
TRIPLET_LD2 = 5
TRIPLET_LD3 = 4
<USER> KO.PY CLIPPING_LEV=
CLIPPING_LEV = 3.33522094727
<USER> KO.PY SHOW_SIGNAL=n
SHOW_SIGNAL = n
number of stars: 222
star 1 of 222

- KO.PY RUNNING                                                         10:28
-                                                                           1
-
-
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
```

D.1 ko.py

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
#
# program by B. Hut
#
# class Result
# def tripletfunction
# def residuals
# def tripletfit
# by M.G.R. Vogelaar, edited by B. Hut
#
# copyright (c) by B. Hut, February 2012
# for the University of Groningen / Kapteyn Astronomical Insitute
# as part of the Bachelor Research Project (KO) of Astronomy

# import modules
# gipsy for gipsy's functionality
from gipsy import *
# matplotlib for plotting
from matplotlib import pyplot as pyplot
# matplotlib: font in plots as in thesis
from matplotlib import rc
rc('text', usetex=True)
# numpy for numerical functions
from numpy import array, where, abs, median, std, delete, exp, sqrt, ones, dot, mean, zeros, arange, pi, convolve
# sys for filename in plots
from sys import argv
# copy for copying arrays
import copy
# datetime for unique output files
import datetime
# scipy.optimize for fitting
from scipy.optimize import leastsq
# pyfits for writing to .fits
import pyfits
# time for monitoring progress
import time
c0 = time.clock() #set clock to zero

# definition of functions

def spec_plot(x, y1, y2, y3, y4):
    #
    #
    # INPUT:
    # x  = array along x axis of plot, wavelength
    # y1 = array of summed spectrum of a star (including central flux)
    # y2 = array of central flux of a star (if inner spaxels do not contain triplet)
    # y3 = array of triplet and continuum fit
    # y4 = array of continuum fit
    #
    # OUTPUT:
    # Plots the summed flux
    pyplot.figure(1)
    ax = pyplot.subplot('111')
    pyplot.suptitle(r'\normalfont Summed spectrum for a star in \verb?%s?' % (s.spec))
    pyplot.annotate(r'\normalfont\small \verb?%s?' % (str(argv[0])),
                    xy=(1.01, 0.98), rotation=90, xycoords='axes fraction', ha='left')
    pyplot.plot(x, y1 ,'k.-', label = r'\normalfont Summed spectrum')
    pyplot.plot(x, y1-y2 ,'g.-', label = r'\normalfont Summed flux (no central spaxels)')
    pyplot.plot(x, y3 ,'b.-', label = r'\normalfont Triplet and continuum fit')
    pyplot.plot(x, y4 ,'r.-', label = r'\normalfont Continuum fit')
    pyplot.grid(True)
    pyplot.xlabel(r'\normalfont Wavelength $\lambda$ (%s)' % (zunit))
    pyplot.ylabel(r'\normalfont Intensity (a.u.)')
    pyplot.legend(loc=0, ncol=1, numpoints = 1, fancybox=True).get_frame().set_alpha(0.7)
    pyplot.show()

def find_stars(image):
    #
    #
    #
    # INPUT
```

```python
    # image          =
    #
    # OUTPUT
    # coordinates  =
    # clipped_data =

    ymin = i.min()
    ymax = i.max()

    clipping_level = 0.02 * ymax
    clipping_level = userint("CLIPPING_LEV=", "Clipping level (range scaled signal = [%s, %s]) [%s]" % (ymin, ymax, clipping_level), defval=clipping_level, default=1)
    cancel("CLIPPING_LEV=")
    anyout(str('CLIPPING_LEV = %s' % (clipping_level)))

    show_signal = 'n'
    show_signal = usertext("SHOW_SIGNAL=", "Show signal and detected stars? y/[n]", defval=show_signal, default=1)
    cancel("SHOW_SIGNAL=")
    anyout(str('SHOW_SIGNAL = %s' % (show_signal)))

    coordinates = []
    clipped_data = []

    # for every row in the image: search for stars, using derivatives
    rownr = 0
    for row in image:
        clipped_row = []
        coordinates_x_row = []

        columnnr = 0
        while columnnr < len(row)-1:
            star = 0
            if row[columnnr] - row[columnnr+1] > 0 and row[columnnr-1] - row[columnnr] < 0:
                column = image[:,columnnr] # corresponding column
                if rownr == len(column)-1:
                    if column[rownr-1] - column[rownr] < 0:
                        # a star detected
                        star = 1
                else:
                    if column[rownr] - column[rownr+1] > 0 and column[rownr-1] - column[rownr] < 0:
                        # a star detected
                        star = 1
            if star == 1 and row[columnnr] < clipping_level:
                # false detection due to clipping level
                star = 0
            clipped_row.append(star)
            if star == 1:
                coordinates.append(array([columnnr, rownr]))
                coordinates_x_row.append(columnnr) # for plotting

            columnnr = columnnr + 1

        # Plotting signal
        if show_signal == 'y':
            global sf
            pyplot.figure(1)
            pyplot.suptitle(r'\normalfont Intensity values for a row in \verb?%s?' % (s.spec))
            xs = row
            ax = pyplot.subplot('111')
            pyplot.annotate(r'\normalfont\small \verb?%s?' % (str(argv[0])),
                            xy=(1.01, 0.98), rotation=90, xycoords='axes fraction', ha='left')
            xi = arange(0, len(xs))
            xmin = min(xi)
            xmax = max(xi)
            pyplot.plot(xi, xs/sf ,'k' )
            for x in coordinates_x_row:
                pyplot.plot([xi[x], xi[x]], [ymin, ymax] ,'b--', label = r'\normalfont\large Star found')
            pyplot.plot([xmin, xmax], [clipping_level/sf, clipping_level/sf] ,'r--', label = r'\normalfont\large Clipping level')
            pyplot.xlim(xmin, xmax)
            pyplot.ylim(ymin/sf, ymax/sf)
            pyplot.grid(True)
            pyplot.xlabel(r'\normalfont\small x (px)')
            pyplot.ylabel(r'\normalfont\small Intensity')
```

```
            pyplot.legend(loc=0, ncol=1, numpoints = 1, fancybox=True).get_frame().
                set_alpha(0.7)
            pyplot.show()

        rownr = rownr + 1
        clipped_data.append(clipped_row)

    return coordinates, clipped_data

def linear_regression( xa, ya ):
    #
    #
    #
    # INPUT
    # xa    =
    # ya    =
    #
    # OUTPUT
    # a     =
    # b     =
    # siga  =
    # sigb  =
    # R     =
    """ Function assumes all weights are equal and
       assumes errors in x much smaller than in y"""
    sumX = sumY = sumXX = sumXY = sumYY = 0.0

    N = len(xa)
    for i in range(N):
        x = xa[i]; y = ya[i]
        sumX += x; sumY += y
        sumXX += x * x; sumYY += y * y
        sumXY += x * y

    sum = N
    delta = sum * sumXX - sumX * sumX
    a = (sumXX*sumY - sumX*sumXY) / delta
    b = (sumXY*sum - sumX*sumY) / delta

    """ Now calculate an estimate for sigma that is equal to all
        data. Use the assumption that the reduced Chi-Square is 1 and
        that sigma^2 is the experimental estimate of the variance in y"""

    degfree = N - 2
    var = (sumYY + a*a*sum +b*b*sumXX - 2.0*(a*sumY + b*sumXY - a*b*sumX) ) / degfree

    siga = sqrt(abs(var*sumXX/delta))
    sigb = sqrt(abs(var*sum/delta))

    R = (sum*sumXY - sumX*sumY) / sqrt(delta*(sum*sumYY-sumY*sumY))

    return( [a, b, siga, sigb, R] )


class Result(object):
    def __init__(self):
        self.params = None

def tripletfunction(x, ie):
    # Given the needed parameters, this function returns the value for intensity =
        continuum(x) + triplet(x)
    # This function calculates the intensity value for given wavelength x and given
        functional parameters.
    # The function is a combination of a linear offset (continuum) and 3 gaussian
        functions (triplet lines)
    #
    # INPUT
    # x      = wavelength positions of the spectrum
    # ie     = initial estimates
    # ie[0]  = amplitude of first gaussian function
    # ie[1]  = mean value of first gaussian function
    # ie[2]  = standard deviation of first gaussian function
    # ie[3]  = amplitude of second gaussian function
    # ie[4]  = mean value of second gaussian function
    # ie[5]  = standard deviation of second gaussian function
    # ie[6]  = amplitude of third gaussian function
    # ie[7]  = mean value of third gaussian function
    # ie[8]  = standard deviation of third gaussian function
    # ie[9]  = slope of linear function
    # ie[10] = intersection of linear function with y axis
```

```
    # mind the global variables ld1, ld2, ld3, which define the line depth ratio
    #
    # OUTPUT
    # intensity = value of the summation of the four functions

    global ld1
    global ld2
    global ld3

    A = ie[0];
    mu1 = ie[1];    sigma1 = ie[2];
    mu2 = ie[3];    sigma2 = ie[4];
    mu3 = ie[5];    sigma3 = ie[6];
    a = ie[7];      b = ie[8];

    gaussian1 = ld1 * A * exp(-(x-mu1)*(x-mu1)/(2.0*sigma1*sigma1))/(sqrt(2*pi)*sigma1)
    gaussian2 = ld2 * A * exp(-(x-mu2)*(x-mu2)/(2.0*sigma2*sigma2))/(sqrt(2*pi)*sigma2)
    gaussian3 = ld3 * A * exp(-(x-mu3)*(x-mu3)/(2.0*sigma3*sigma3))/(sqrt(2*pi)*sigma3)
    linear    = x*(ones(len(x))*a) + b
    intensity = gaussian1 + gaussian2 + gaussian3 + linear

    return(intensity)

def tripletfunction_error(x, ie, std):
    # met onder andere, waarom err_sigma1 = 0.06? (= resolution element)
    # en ook assumed no error in lambda0
    #
    # INPUT
    # x     = wavelength positions of the spectrum
    # ie    = initial estimates as in def tripletfunction
    # std   =
    #
    # mind the global variables ld1, ld2, ld3, which define the line depth ratio
    #
    # OUTPUT
    # intensity_error = error in the intensity as an array

    global ld1
    global ld2
    global ld3

    A = ie[0];
    mu1 = ie[1];    sigma1 = abs(ie[2]); #negative sigma not possible
    mu2 = ie[3];    sigma2 = abs(ie[4]); #negative sigma not possible
    mu3 = ie[5];    sigma3 = abs(ie[6]); #negative sigma not possible
    a = ie[7];      b = ie[8];

    dIdA  = ld1 * exp(-(x-mu1)*(x-mu1)/(2.0*sigma1*sigma1))/(sqrt(2*pi)*sigma1) + ld2 *
        exp(-(x-mu2)*(x-mu2)/(2.0*sigma2*sigma2))/(sqrt(2*pi)*sigma2) + ld3 * exp(-(x-
        mu3)*(x-mu3)/(2.0*sigma3*sigma3))/(sqrt(2*pi)*sigma3)
    dIdsigma1 = ld1 * A * (sigma1**2.-2*(x-mu1)**2.)/(2.*sqrt(2.*pi)*sigma1**(7./2.)) *
        exp(-(x-mu1)*(x-mu1)/(2.0*sigma1*sigma1))/(sqrt(2.*pi)*sigma1)
    dIdsigma2 = ld2 * A * (sigma2**2.-2*(x-mu2)**2.)/(2.*sqrt(2.*pi)*sigma2**(7./2.)) *
        exp(-(x-mu2)*(x-mu2)/(2.0*sigma2*sigma1))/(sqrt(2.*pi)*sigma2)
    dIdsigma3 = ld3 * A * (sigma3**2.-2*(x-mu3)**2.)/(2.*sqrt(2.*pi)*sigma3**(7./2.)) *
        exp(-(x-mu3)*(x-mu3)/(2.0*sigma3*sigma3))/(sqrt(2.*pi)*sigma3)

    err_A = std*ones(len(x))
    err_sigma1 = 0.06*ones(len(x))
    err_sigma2 = 0.06*ones(len(x))
    err_sigma3 = 0.06*ones(len(x))

    err_intensity = sqrt(abs((dIdA*err_A)**2. + (dIdsigma1*err_sigma1)**2. + (dIdsigma2*
        err_sigma2)**2. + (dIdsigma2*err_sigma2)**2.))

    return err_intensity

def linefunction(x, ie):
    #
    #
    #
    # INPUT
    # x         =
    # ie        = initial estimates
    #
    # OUTPUT
    # intensity =
    A = ie[0];
    mu = ie[1];
    sigma = ie[2];
```

```
    a = ie[3];
    b = ie[4];
    gaussian = A * exp(-(x-mu)*(x-mu)/(2.0*sigma*sigma))/(sqrt(2*pi)*sigma)
    linear   = x*(ones(len(x))*a) + b
    intensity = linear + gaussian

    return(intensity)

def continuumfunction(x, ie):
    #
    #
    #
    # INPUT
    # x          =
    # ie         = initial estimates
    #
    # OUTPUT
    # intensity =
    a = ie[0];
    b = ie[1];
    intensity = x*(ones(len(x))*a) + b

    return(intensity)

def residuals(p, y, x):
    # Calculates the difference between the data and the fitted function
    #
    #
    # INPUT
    # p = parameters that are input for tripletfunction. So the initial estimates.
    # y = array that contains the data
    # x = array that contains the x values, which is passed on to tripletfuncion
    #
    # OUTPUT
    # y - tripletfunction(x,p)  = array that contains the difference
    return y - tripletfunction(x, p)

def tripletfit(x, y, initialestimates):
    # This function fits the function to the data.
    #
    # The formal 1-sigma errors in each parameter, computed from the covariance matrix.
    #  If a parameter is held fixed, or if it touches a boundary, then the error is
        reported as zero.
    #
    # If the fit is unweighted (i.e. no errors were given, or the weights were uniformly
         set to unity),
    #  then .perror will probably not represent the true parameter uncertainties.
    #
    #  *If* you can assume that the true reduced chi-squared value is unity
    #   --meaning that the fit is implicitly assumed to be of good quality--
    #   then the estimated parameter uncertainties can be computed by scaling .perror by
         the measured chi-squared value.
    #
    #           dof = len(x) - len(result.params) # deg of freedom
    #           # scaled uncertainties
    #           pcerror = result.perror * numpy.sqrt(result.fnorm / dof)
    #
    # Usage of result:
    # result = tripletfit(x, y, p0)
    # if result.params == None:
    #     print 'error message = ', result.errmsg
    # else:
    #     print "Iterations: ", result.niter
    #     print "Fitted pars: ", result.params
    #     print "Uncertainties assuming chi-sqr = 1.0: ", result.pcerror
    #
    #
    # INPUT xi xs p0
    # x                  = wavelength positions of the spectrum
    # y                  = flux of the spectrum (intensities)
    # initialestimates = initial estimates for the constants in the to be fitted
        function
    #                    see documentation of function 'tripletfunction'
    #
    # OUTPUT
    # see 'Usage of result', above

    par, cov_x, infodict, mesg, ier = leastsq(residuals, initialestimates, args=(y, x),
        full_output=1)
    result = Result()
```

```
    if ier < 1 or ier > 4:
        result.params = None
        return result
    chi = infodict['fvec']**2
    if not cov_x is None: #try:
        result.fnorm = chi.sum()
        result.params = par
        result.perror = sqrt(cov_x.diagonal())
        result.errmsg = mesg
        result.niter = infodict['nfev']
        dof = len(x) - len(result.params) # deg of freedom
        # Uncertainties for unweighted data
        result.pcerror = result.perror * sqrt(result.fnorm/dof)
    else: #except:
        result.params = None

    return result

def find_triplet(flux, xi, mu1, d2, d3, ld1, ld2, ld3):
    # Selects the position of the triplet lines
    # This function looks for the triplet somewhere in the spectrum (flux & xi) after lambda = mu1.
    # The triplet is characterized by d2, d3, ld1, ld2, ld3.
    #
    # INPUT
    # flux = intensities of the spectrum
    # xi   = wavelength positions of the spectrum
    # mu1  = lower bound of position in wavelength of first line
    # d2   = difference in wavelength between first and second line in nm
    # d3   = difference in wavelength between first and third line in nm
    # ld1  = line depth ratio (first line). ld1 : ld2 : ld3
    # ld2  = line depth ratio (second line)
    # ld3  = line depth ratio (third line)
    #
    # OUTPUT
    # triplet_I = intensities of the triplet
    # triplet_x = wavelengths of the triplet

    xi = array(xi)
    flux = array(flux)

    tol = 0.5
    num_std = 2

    # get spectrum for wavelengths > mu1 for redshifted triplet (redshift --> mu1 is under bound)
    x_r = copy.copy(xi)
    f_r = copy.copy(flux)
    x_r = array(x_r[where(x_r > mu1)])
    f_r = f_r[len(f_r)-len(x_r):len(f_r)]

    continuum = median(f_r)
    std = numpy.std(f_r)

    not_found_r = 0
    i = 0
    inc = x_r[1]-x_r[0]
    while True:
      mu1_r = continuum - f_r[i]
      mu2_r = continuum - f_r[i + d2/inc]
      mu3_r = continuum - f_r[i + d3/inc]
      if abs(mu1_r) > num_std*std: # test that the selected line is significant larger than the standard deviation
          if abs(mu1_r/ld1*ld2) > abs(mu2_r*(1-tol)) and abs(mu1_r/ld1*ld2) < abs(mu2_r*(1+tol)) and abs(mu1_r/ld1*ld3) > abs(mu3_r*(1-tol)) and abs(mu1_r/ld1*ld3) < abs(mu3_r*(1+tol)): # if triplet found: stop
                                                  # first 2 conditions: first line related to second line
                                                  # second 2 conditions: first line related to third line
              break
      i = i + 1
      if i + d3/inc > len(f_r) - 1:
          mu1_r = 0;  mu2_r = 0;  mu3_r = 0;
          not_found_r = 1
          break

    triplet_redshifted_I = array([mu1_r, mu2_r, mu3_r])
    triplet_redshifted_x = array([x_r[i], x_r[i + d2/inc], x_r[i + d3/inc]])

```

```
    # get spectrum for wavelengths > mu1 for blueshifted triplet (blueshift --> mu1 is upper bound)
    x_b = copy.copy(xi)
    f_b = copy.copy(flux)
    x_b = array(x_b[where(x_b < (mu1 + d3))]) # account for the width of the whole triplet
    f_b = f_b[0:len(x_b)]

    continuum = median(f_b)
    std = numpy.std(f_b)

    not_found_b = 0
    i = len(f_b) - 1 - d3/inc
    while True:
      mu1_b = continuum - f_b[i]
      mu2_b = continuum - f_b[i + d2/inc]
      mu3_b = continuum - f_b[i + d3/inc]
      if abs(mu1_b) > num_std*std: # test that the selected line is significant larger than the standard deviation
          if abs(mu1_b/ld1*ld2) > abs(mu2_b*(1-tol)) and abs(mu1_b/ld1*ld2) < abs(mu2_b*(1+tol)) and abs(mu1_b/ld1*ld3) > abs(mu3_b*(1-tol)) and abs(mu1_b/ld1*ld3) < abs(mu3_b*(1+tol)): # if triplet found: stop
                                                  # first 2 conditions: first line related to second line
                                                  # second 2 conditions: first line related to third line
              break
      i = i - 1
      if i < 0:
          mu1_b = 0;  mu2_b = 0;  mu3_b = 0;
          not_found_b = 1
          break

    triplet_blueshifted_I = array([mu1_b, mu2_b, mu3_b])
    triplet_blueshifted_x = array([x_b[i], x_b[i + d2/inc], x_b[i + d3/inc]])

    if abs(mu1 - triplet_blueshifted_x[0]) > abs(mu1 - triplet_redshifted_x[0]):
        triplet_I = triplet_redshifted_I
        triplet_x = triplet_redshifted_x
    elif not_found_b == 1:
        triplet_I = triplet_redshifted_I
        triplet_x = triplet_redshifted_x
    elif not_found_r == 1:
        triplet_I = triplet_blueshifted_I
        triplet_x = triplet_blueshifted_x
    elif not_found_b == 1 and not_found_r == 1:
        return False
    else:
        triplet_I = triplet_blueshifted_I
        triplet_x = triplet_blueshifted_x

    return triplet_I, triplet_x

def snr_estimation_by_region(flux, triplet, x, xmax, xmin):
    # Estimates signal-to-noise ratio on a specified region
    # This function estimates the SNR on a user specified region, using the median of the fitted function as
    #   signal and the standard deviation of the flux as noise.
    # The minimal and maximum values for wavelength given by the user, define the region for SNR estimation
    #
    # INPUT
    # flux      = intensity data of the spectrum
    # triplet   = fitted triplet AND continuum (--> signal)
    # x         = wavelength positions
    # xmax      = wavelength max for SNR estimation
    # xmin      = wavelength min for SNR estimation
    #
    # OUTPUT
    # snr = calculated value for the SNR
    # x   = used wavelength positions for SNR estimation

    flux = array(flux)

    x_snr = copy.copy(x)
    triplet_snr = copy.copy(triplet)

    # select boundaries of specified region
    indexmin = 0
    indexmax = len(x)
```

```
    while True:
        if x_snr[0]>xmin:
            break
        del x_snr[0]
        indexmin = indexmin+1
    while True:
        if x_snr[-1]<xmax:
            break
        del x_snr[-1]
        indexmax = indexmax - 1

    x_snr = x[indexmin:indexmax] # only select specified region
    flux = flux[indexmin:indexmax] # only select specified region
    triplet_snr = triplet_snr[indexmin:indexmax] # only select specified region
    signal = numpy.median(triplet_snr)
    std = numpy.std(flux)

    snr = signal / std
    return snr, x_snr

def equivalent_widths(xi, ie, std_res = 0):
    #
    #
    # INPUT
    # xi      =
    # ie      =
    # std_res =
    #
    # OUTPUT
    # EW1     =
    # EW2     =
    # EW3     =
    # err_EW1 =
    # err_EW2 =
    # err_EW3 =

    line1 = linefunction(xi, [ie[0], ie[1], ie[2], ie[7], ie[8]])
    line2 = linefunction(xi, [ie[0], ie[3], ie[4], ie[7], ie[8]])
    line3 = linefunction(xi, [ie[0], ie[5], ie[6], ie[7], ie[8]])
    continuum = continuumfunction(xi, [ie[7], ie[8]])

    integral1 = 1 - line1/continuum
    integral2 = 1 - line2/continuum
    integral3 = 1 - line3/continuum

    EW1 = sum(integral1)
    EW2 = sum(integral2)
    EW3 = sum(integral3)

    line1_max  = linefunction(xi, [ie[0]+std_res, ie[1], ie[2], ie[7], ie[8]])
    line1_min = linefunction(xi, [ie[0]-std_res, ie[1], ie[2], ie[7], ie[8]])
    line2_max  = linefunction(xi, [ie[0]+std_res, ie[3], ie[4], ie[7], ie[8]])
    line2_min = linefunction(xi, [ie[0]-std_res, ie[3], ie[4], ie[7], ie[8]])
    line3_max  = linefunction(xi, [ie[0]+std_res, ie[5], ie[6], ie[7], ie[8]])
    line3_min = linefunction(xi, [ie[0]-std_res, ie[5], ie[6], ie[7], ie[8]])

    integral1_min = 1 - line1_min/continuum
    integral1_max = 1 - line1_max/continuum
    integral2_min = 1 - line2_min/continuum
    integral2_max = 1 - line2_max/continuum
    integral3_min = 1 - line3_min/continuum
    integral3_max = 1 - line3_max/continuum

    err_EW1min = abs(sum(integral1_min) - EW1)
    err_EW1max = abs(sum(integral1_max) - EW1)
    err_EW2min = abs(sum(integral2_min) - EW2)
    err_EW2max = abs(sum(integral2_max) - EW2)
    err_EW3min = abs(sum(integral3_min) - EW3)
    err_EW3max = abs(sum(integral3_max) - EW3)

    err_EW1 = (err_EW1max + err_EW1min)/2.
    err_EW2 = (err_EW2max + err_EW2min)/2.
    err_EW3 = (err_EW3max + err_EW3min)/2.

    return EW1, EW2, EW3, err_EW1, err_EW2, err_EW3

def export_csv(data):
    #
    #
    # INPUT
```

```
    # data     =
    #
    # OUTPUT
    # str(filename) =
    filename = 'muse_ko_'+str(datetime.datetime.now().isoformat())+'.csv'
    f = open(filename, "w")
    for value in data:
        f.write(str(value)+',')
    f.close()
    return str(filename)

def export_spec_csv(legend, datacube, ID):
    #
    #
    # INPUT
    # data     =
    #
    # OUTPUT
    # str(filename) =
    filename = 'muse_spec_'+str(ID)+'.csv'
    f = open(filename, "w")
    f.write('# ')
    for column in legend:
        f.write(str(column)+',')
    f.write('\n')

    for i in arange(len(datacube[0])):
        for data in datacube:
            f.write(str(data[i])+',')
        f.write('\n')
    f.close()

    return str(filename)

def x_to_RA(x):
    # Calculates the corresponding celestiol long. of x pixel position, using header info
    #
    # INPUT
    # x = x coordinate in pixels
    #
    # mind the global variables crpixx, crvalx, cdeltx
    #
    # OUTPUT
    # RA = RA coordinate in units of header
    global crpixx
    global crvalx
    global cdeltx
    RA = (x - crpixx) * cdeltx + crvalx
    return RA

def y_to_DEC(y):
    # Calculates the corresponding celestiol lat. of y pixel position, using header info
    #
    # INPUT
    # y = y coordinate in pixels
    #
    # OUTPUT
    # DEC = DEC coordinate in units of header
    global crpixy
    global crvaly
    global cdelty
    DEC = (y - crpixy) * cdelty + crvaly
    return DEC

def count_to_mag(count):
    # Calculates instrumental magnitude out of number of photon count
    #
    # INPUT
    # counts = number of counts
    #
    # OUTPUT
    # mag     = corresponding instrumental magnitude
    mag = -2.5 * np.log10(count)
    return mag

def nm_to_angstrom(nm):
    # Calculates angstroms for a given number of nanometers
    #
    # INPUT
```

```
    # nm       = nanometers to be converted
    #
    # OUTPUT
    # angstrom = corresponding value in angstrom
    angstrom = nm * 10.
    return angstrom

# initialize task communication with GIPSY's control process (Hermes)
init()

# while input of new set
while True:
    input = usertext("SET=", "Set, (subsets)  [empty -> quit]", defval='',default=1)
    if input=='':   # to end while loop, using usertexts default
        break

    cancel("SET=")  # make sure user can be prompted again
    s = Set(input)  # set or subset specification (also for FITS files, see:
                    # http://www.intra.astro.rug.nl/~gipsy/pguide/pygipsy.html )

    i = s.image           # retrieve data
    sf = 1/max(i[0,0,:])  # pick a scale factor
    i = i * sf            # scale to make sure computer can handle all values

    # Retrieve header data
    axes_ctype = []
    axes_units = []
    axes_naxis = []
    axes_cdelt = []
    axes_crval = []
    axes_crpix = []
    for ax in s.axes():
        axes_ctype.append(ax.ctype)
        axes_units.append(ax.cunit)
        axes_cdelt.append(ax.cdelt)
        axes_crval.append(ax.crval)
        axes_crpix.append(ax.crpix)
    lo, hi = s.range(0)
    for n in range(s.naxis):
        axes_naxis.append(s.grid(n, hi)-s.grid(n, lo))
    xname = axes_ctype[0]
    yname = axes_ctype[1]
    zname = axes_ctype[2]
    xunit = axes_units[0]
    yunit = axes_units[1]
    zunit = axes_units[2]
    naxisx = axes_naxis[0]
    naxisy = axes_naxis[1]
    naxisz = axes_naxis[2]
    cdeltx = axes_cdelt[0]
    cdelty = axes_cdelt[1]
    cdeltz = axes_cdelt[2]
    crvalx = axes_crval[0]
    crvaly = axes_crval[1]
    crvalz = axes_crval[2]
    crpixx = axes_crpix[0]
    crpixy = axes_crpix[1]
    crpixz = axes_crpix[2]

    spectra_output = 'n'
    spectra_output = usertext("SPECTRA_OUTPUT=", "Do you want to output summed spectra as CSV? y/[n]", defval=spectra_output,default=1)
    cancel("SPECTRA_OUTPUT=")
    anyout(str('SPECTRA_OUTPUT = '+spectra_output))

    xi = ones(naxisz)+crvalz+array(range(0,naxisz))*cdeltz
    xi = xi.tolist()

    xmin = xi[0]
    xmax = xi[-1]
    xmin = userint("MINX=", "Minimum %s coordinate  [%d]" % (zname,xmin), defval=xmin, default=1)
    xmax = userint("MAXX=", "Maximum %s coordinate  [%d]" % (zname,xmax), defval=xmax, default=1)
    cancel("MINX=")
    cancel("MAXX=")
    anyout(str('MINX = %s' % (xmin)))
    anyout(str('MAXX = %s' % (xmax)))

    # delete part that will not be used
```

```
    indexmin = 0
    indexmax = len(xi)
    cinmin = indexmin
    cinmax = indexmax
    while True:
        if xi[0]>xmin:
            break
        del xi[0]
        indexmin = indexmin+1
    while True:
        if xi[-1]<xmax:
            break
        del xi[-1]
        indexmax = indexmax - 1

    # select region of snr estimation
    snr_xmin = xi[0]
    snr_xmax = xi[-1]
    snr_xmin = userint("SNRMINX=", "Minimum %s coordinate for SNR estimation  [%d]" % (zname,snr_xmin), defval=snr_xmin, default=1)
    snr_xmax = userint("SNRMAXX=", "Maximum %s coordinate for SNR estimation  [%d]" % (zname,snr_xmax), defval=snr_xmax, default=1)
    cancel("SNRMINX=")
    cancel("SNRMAXX=")
    anyout(str('SNRMINX = %s' % (xmin)))
    anyout(str('SNRMAXX = %s' % (xmax)))

    # user has to specify some characeristics of the triplet
    # default characeristics of NR CaT at rest
    mu1 = 849.8
    mu2 = 854.2
    mu3 = 866.2
    ld1 = 3
    ld2 = 5
    ld3 = 4
    mu1 = userint("TRIPLET_MU1=", "Rest position of first triplet line in nm [%d]" % (mu1), defval=mu1, default=1)
    mu2 = userint("TRIPLET_MU2=", "Rest position of second triplet line in nm [%d]" % (mu2), defval=mu2, default=1)
    mu3 = userint("TRIPLET_MU3=", "Rest position of third triplet line in nm [%d]" % (mu3), defval=mu3, default=1)
    ld1 = userint("TRIPLET_LD1=", "Gaussian line depth ratio (first line) [%d]" % (ld1), defval=ld1, default=1)
    ld2 = userint("TRIPLET_LD2=", "Gaussian line depth ratio (second line) [%d]" % (ld2), defval=ld2, default=1)
    ld3 = userint("TRIPLET_LD3=", "Gaussian line depth ratio (third line) [%d]" % (ld3), defval=ld3, default=1)
    anyout(str('TRIPLET_MU1 = %s' % (mu1)))
    anyout(str('TRIPLET_MU2 = %s' % (mu2)))
    anyout(str('TRIPLET_MU3 = %s' % (mu3)))
    anyout(str('TRIPLET_LD1 = %s' % (ld1)))
    anyout(str('TRIPLET_LD2 = %s' % (ld2)))
    anyout(str('TRIPLET_LD3 = %s' % (ld3)))
    cancel("TRIPLET_MU1=")
    cancel("TRIPLET_MU2=")
    cancel("TRIPLET_MU3=")
    cancel("TRIPLET_LD1=")
    cancel("TRIPLET_LD2=")
    cancel("TRIPLET_LD3=")

    show_flux = 'n'
    show_flux = usertext("SHOW_FLUX=", "Show summed flux and fit of triplet of each star ? y/[n]", defval=show_flux, default=1)
    cancel("SHOW_FLUX=")
    anyout(str('SHOW_FLUX = %s' % (show_flux)))


    # Calculate other characteristics of CaT from input
    A1 = -1
    A2 = A1/ld1*ld2
    A3 = A1/ld1*ld3
    sigma1 = 0.3
    sigma2 = sigma1
    sigma3 = sigma1
    d2 = mu2 - mu1 # separation between first and second triplet line
    d3 = mu3 - mu1 # separation between first and third triplet line

    #find central coordinates of stars in image at first wavelength position
    result = find_stars(i[0,:,:])
    coordinates = result[0]
```

```
    clipped_data = result[1]
    number_of_stars = len(coordinates)
    anyout(str('number of stars: '+str(number_of_stars)))

    # declare variables for star-iterative process
    star_nr = 1
    sX_data = []
    sY_data = []
    sRA_data = []
    sDEC_data = []
    sboxradius_data = []
    slambda1_data = []
    sEW1_data = []
    sEW2_data = []
    sEW3_data = []
    serr_EW1_data = []
    serr_EW2_data = []
    serr_EW3_data = []
    sSNR_data = []
    sBcounts_data = [] # these counts are pseudo-B
    sVcounts_data = [] # these counts are pseudo-V
    sNr_data = []

    # begin iteration over all star coordinates
    for coordinate in coordinates:
        # set variables of star
        anyout(str('star '+str(star_nr)+' of '+str(number_of_stars)))
        xstar = coordinate[0]
        ystar = coordinate[1] + 1
        offset = 0
        summed_flux_star = zeros(len(xi))     # len(xi) = len(xs) but xs not yet defined
        summed_flux_centrum = zeros(len(xi)) # len(xi) = len(xs) but xs not yet defined

        # begin iteration over (box-shaped) shells around star
        while offset != -1:
            # calculate coordinates of pixels in box around star
            # (inner box will be deleted in a upcoming for loop to create the shell)
            xp_lo = xstar - min(offset, xstar)                        # lower x coordinate of shell
            xp_hi = xstar + min(offset, len(i[0,:,0]) - xstar - 1)  # upper x coordinate of shell
            yp_lo = ystar - min(offset, ystar)                        # lower y coordinate of shell
            yp_hi = ystar + min(offset, len(i[0,0,:]) - ystar - 1)  # upper y coordinate of shell

            # begin iteration over all x values (bounded by xp_lo and xp_hi) in shell
            for xp in arange(xp_lo, xp_hi+1):
                # set/declare variables as empty
                im_row = []

                # begin iteration over all y values (bounded by yp_lo and yp_hi) in column
                for yp in arange(yp_lo, yp_hi+1):
                    # check if pixel is really in shell or in inner box (if in inner box: go to next yp value)
                    if ((xp == xp_lo) or (xp == xp_hi) or (yp == yp_lo) or (yp == yp_hi)) == False:
                        continue

                    # get spectrum for position (xp, yp):
                    xs = i[:,xp,yp]

                    # remove useless part (setted by user)
                    xs = xs[indexmin:indexmax]
                    if (xp == xp_lo and yp == yp_lo):
                        summed_flux_shell = zeros(len(xs))

                    # linear approximation of continuum using least squares method for initial estimates of triplet & continuum fit
                    a, b, sigA, sigB, R = linear_regression(xi, xs)
                    yl = b*array(xi)+a

                    # Find the triplet, using its characeristics
                    result = find_triplet(xs, xi, mu1, d2, d3, ld1, ld2, ld3)
                    if result != False:
                        t_I, t_x = result
                    else:
                        break

```

```
                    # triplet fitting | initial estimates, using location of triplet in data (suffix '_d')
                    mu1_d = t_x[0];           mu2_d = mu1_d + d2;      mu3_d = mu1_d + d3
                    ie = [A1, mu1_d, sigma1, mu2_d, sigma2, mu3_d, sigma3, a, b]

                    # actual fitting
                    result = tripletfit(xi, xs, ie)
                    if result.params == None: # if fitting has failed
                        snr = 'NaN'
                    else:
                        ytc = tripletfunction(xi, result.params) # curve containing triplet and continuum
                        a = result.params[7]
                        b = result.params[8]
                        yc = continuumfunction(xi, [a,b]) # linear (continuum) part of (triplet & continuum) curve\
                        residual = xs-ytc
                        std_res = eval(str(numpy.std(residual)))
                        err_ytc = tripletfunction_error(xi, result.params, std_res/5)

                        # snr estimation
                        snr, xi_snr = snr_estimation_by_region(xs, ytc, xi, snr_xmax, snr_xmin)

                        # sum flux
                        summed_flux_shell = summed_flux_shell + xs
                    #last line in iteration over all y values in column
                #continue iteration over all x values in shell

                #last line in iteration over all x values in shell
            #continue iteration over shells

            # look for triplet in summed spectrum of shell (using rest frame characteristics)
            result = find_triplet(summed_flux_shell, xi, mu1, d2, d3, ld1, ld2, ld3)
            if result != False:
                t_I, t_x = result
            else:
                anyout(str('No triplet found in summed spectrum of shell!'))
                anyout(str('Fitting not possible in summed spectrum of shell!'))
                anyout(str('Skip this potential star and go to next.'))
                break

            # triplet fitting | initial estimates, using location of triplet in summed spectrum of shell (suffix '_d')
            mu1_d = t_x[0];           mu2_d = mu1_d + d2;      mu3_d = mu1_d + d3
            ie = [A1, mu1_d, sigma1, mu2_d, sigma2, mu3_d, sigma3, a, b]
            result = tripletfit(xi, summed_flux_shell, ie)

            if result.params == None: # if fitting failed
                anyout(str('Triplet found in summed spectrum of shell.'))
                anyout(str('No fit in summed spectrum of shell!'))
                anyout(str('Summing next square (r -> r+2).'))
                snr_shell = 0.
            else: # if fitting succeeded
                # retrieve curves of triplet and continuum (and continuum only) and do snr estimation of summed spectrum in shell
                ytc_shell = tripletfunction(xi, result.params) # curve containing triplet and continuum
                snr_shell, xi_snr_shell = snr_estimation_by_region(summed_flux_shell, ytc_shell, xi, snr_xmax, snr_xmin)
                snr_star, xi_snr_star = snr_estimation_by_region(summed_flux_star, ytc_shell, xi, snr_xmax, snr_xmin)

            # check whether snr of shell is bigger than snr of star
            if offset == 0: # central pixel of star is current pixel
                snr_star = 0. # to force that this pixel will be used in the summation
                last_snr_shell = 0.
                summed_flux_star = zeros(len(xs))

            # make sure that the initial summed_flux_star = zeros(len(xi)) can be replaced by summed_flux_shell
            if snr_star == 'inf':
                snr_star = 0.

            # make sure that the last_snr_shell will not become infinite (otherwise the script will enter a loop of inf length)
            if last_snr_shell == 'inf':
                last_snr_shell = 0.

```

```
            if snr_shell > last_snr_shell: # if snr of shell is higher than snr of pixels of star: add flux
                summed_flux_star = summed_flux_star + summed_flux_shell
                last_snr_shell = snr_shell # remember snr of this shell

            elif summed_flux_star.any() == zeros(len(xi)).any() and offset < 10:
                summed_flux_star = summed_flux_star + summed_flux_shell
                summed_flux_centrum = summed_flux_centrum + summed_flux_shell
                anyout(str('Force new shell because inner pixels have bad spectrum'))
            else:
                sboxradius_data.append(offset -0.0)
                summed_flux_star = summed_flux_star
                offset = -2

            # prepare variables in iteration to go to next shell
            # reset summed flux of shell
            summed_flux_shell = zeros(len(xs))
            offset = offset + 1
            #last line in iteration over shell
        #continue iteration over all star coordinates

        # calculate characteristics of stars, by fitting a triplet (again)
        # to measure equivalent width, the summed flux of the center is subtracted. summed_flux_centrum is nonzero when there is no (complete) triplet in the stars (x,y)center only. It is not subtracted when measuring flux.

        # linear approximation of continuum using least squares method for initial estimates of triplet & continuum fit
        a, b, sigA, sigB, R = linear_regression(xi, summed_flux_star-summed_flux_centrum)
        yl = b*array(xi)+a

        # find the triplet, using its characeristics
        result = find_triplet(summed_flux_star-summed_flux_centrum, xi, mu1, d2, d3, ld1, ld2, ld3)
        if result != False:
            t_I, t_x = result
        else:
            break

        # triplet fitting | initial estimates, using location of triplet in data (suffix '_d')
        mu1_d = t_x[0];          mu2_d = mu1_d + d2;     mu3_d = mu1_d + d3
        ie = [A1, mu1_d, sigma1, mu2_d, sigma2, mu3_d, sigma3, a, b]

        # actual fitting
        result = tripletfit(xi, summed_flux_star-summed_flux_centrum, ie)
        if result.params == None: # if fitting has failed
            EW1 = 'NaN'
            EW2 = 'NaN'
            EW3 = 'NaN'
            lambda1 = 'NaN'
            Bcounts = 'NaN'
            Vcounts = 'NaN'
            snr = 'NaN'
            lambda1 = 'NaN'
        else:
            ytc = tripletfunction(xi, result.params) # curve containing triplet and continuum
            a = result.params[7]
            b = result.params[8]
            yc = continuumfunction(xi, [a,b]) # linear (continuum) part of (triplet & continuum) curve

            # Show flux and fit to it
            if show_flux == 'y':
                spec_plot(xi, summed_flux_star, summed_flux_centrum, ytc, yc)

            # Equivalent width calculation
            residual = xs - ytc
            std_res = numpy.std(residual)
            EW1, EW2, EW3, err_EW1, err_EW2, err_EW3 = equivalent_widths(xi, result.params, std_res)

            # snr estimation
            snr, xi_snr = snr_estimation_by_region(summed_flux_star, ytc, xi, snr_xmax, snr_xmin)

            EW1 = nm_to_angstrom(EW1)
            if EW1 < 0.:
```

```
                EW1 = 'NaN'
            EW2 = nm_to_angstrom(EW2)
            if EW2 < 0.:
                EW2 = 'NaN'
            EW3 = nm_to_angstrom(EW3)
            if EW3 < 0.:
                EW3 = 'NaN'
            Bcounts = sum(summed_flux_star[0:len(summed_flux_star)/2])
            Vcounts = sum(summed_flux_star[len(summed_flux_star)/2:len(summed_flux_star)])
            lambda1 = nm_to_angstrom(result.params[1])

        sEW1_data.append(EW1)
        serr_EW1_data.append(err_EW1)
        sEW2_data.append(EW2)
        serr_EW2_data.append(err_EW2)
        sEW3_data.append(EW3)
        serr_EW3_data.append(err_EW3)
        sX_data.append(xstar)
        sY_data.append(ystar)
        sRA_data.append(x_to_RA(xstar))
        sDEC_data.append(y_to_DEC(ystar))
        slambda1_data.append(lambda1)
        sSNR_data.append(snr)
        sBcounts_data.append(Bcounts)
        sVcounts_data.append(Vcounts)
        sNr_data.append(star_nr)

        # output summed spectrum of star with number star_nr (setted by user)
        if spectra_output == 'y':
          ID = str(star_nr)
          legend = array(['Spectral_Value', 'Flux_Value'])
          datacube = array([xi, summed_flux_star/sf]) # rescale wavelength output
          anyout(str('Output summed spectrum: '+export_spec_csv(legend, datacube, ID)))
        star_nr = star_nr + 1
        #last line in iteration over all star coordinates
    #continue part outside any iteration

    # rescale output that is effected by the initial scaling
    sBcounts_data = sBcounts_data / sf
    sVcounts_data = sVcounts_data / sf

    # generate output
    anyout(str('Output generated:'))
    anyout(str('X       = "%s"' % (export_csv(sX_data))))
    anyout(str('Y       = "%s"' % (export_csv(sY_data))))
    anyout(str('RA      = "%s"' % (export_csv(sRA_data))))
    anyout(str('DEC     = "%s"' % (export_csv(sDEC_data))))
    anyout(str('boxrad  = "%s"' % (export_csv(sboxradius_data))))
    anyout(str('lambda1 = "%s"' % (export_csv(slambda1_data))))
    anyout(str('EW1     = "%s"' % (export_csv(sEW1_data))))
    anyout(str('EW2     = "%s"' % (export_csv(sEW2_data))))
    anyout(str('EW3     = "%s"' % (export_csv(sEW3_data))))
    anyout(str('EW1err  = "%s"' % (export_csv(serr_EW1_data))))
    anyout(str('EW2err  = "%s"' % (export_csv(serr_EW2_data))))
    anyout(str('EW3err  = "%s"' % (export_csv(serr_EW3_data))))
    anyout(str('SNR     = "%s"' % (export_csv(sSNR_data))))
    anyout(str('Vcounts = "%s"' % (export_csv(sVcounts_data))))
    anyout(str('Bcounts = "%s"' % (export_csv(sBcounts_data))))
    anyout(str('num     = "%s"' % (export_csv(sNr_data))))

    # close the set
    del s

# terminate task execution
finis()
```