FancyCluster: a Tool for Realistic Visualization of Stellar Clusters

Computational Astrophysics - Final Report

Izumi Takimoto Schmiegelow
Utsav Bose

May 2023

Abstract

We present `FancyCluster`, a tool to simulate realistic images of star clusters. The simulation uses the luminosity and temperature of the stars to predict the R, V and B magnitudes and creates an image using these as RGB colours. The algorithm was written with the goal of creating images which resemble pictures taken with a telescope. We take into account physical effects such as diffraction in the instrument, creating spikes on bright stars, and combine these with other graphical elements which are not optics-based but were added to improve the realism of the final image.

1 Introduction

Graphical representation has always been an essential way to present scientific data. This is usually done in the form of plots with representative symbols instead of a realistic image. Creating a realistic image is a complex task, requiring a great amount of time.

Here, we present the `FancyCluster` Python library, a tool to take the visualization of star clusters a step further, with a full simulation of a telescope picture based on data containing the positions, the luminosities and the temperatures of the stars. This can be used for simulations of future arrangements of the star cluster when the positions are predicted based on current data. It can also be used to simulate current data with the advantage of not having background stars like in telescope pictures, so we can clearly see which stars are part of the cluster.

We make an estimation of the RGB colours of each star and take into account the optical effects present in images taken by reflective telescopes. For making the simulation we based ourselves on images taken by some of the best telescopes in use to day such as the Hubble Space Telescope and the James Webb Space Telescope and aimed at creating images of the stars with a similar shape to what

is observed. For this, we combined both physical laws and graphical elements added as an artistic choice to increase the similarity to telescope pictures.

This paper is organized as follows: In Section 2 the structure and contents of the required input file are described. In Section 3 the conversion from the coordinate positions in the input file to the pixel position in the frame is explained, Section 4 contains the mathematical steps used to simulate the colours of the stars, Section 5 contains the steps for modelling and creating the shape of the stars in the pictures. The discussion can be found in Section 6 and a short conclusion in Section 7.

2 Input Files

The standard input file is a .txt file containing 10 columns with information about each star. The columns correspond respectively to the time, the mass, the position in the x, y and z axes, the velocity in the x, y and z axes, the temperature and the luminosity. For this version of `FancyCluster` only the luminosity, the temperature and the x and y positions are used. The luminosity and temperature are later converted into B, V and R magnitudes (4). The positions are given in units of arcseconds from the centre of the cluster. The temperature is given in Kelvin and the luminosity in solar units.

2.1 SIMBAD Files

We included the option to use datasets of clusters from the SIMBAD [3] website to generate the images. The SIMBAD database contains data from various surveys that studied the same object and allows users to download a file with all the information combined directly from the website.

The SIMBAD files are different from the standard input file. They are Ascii files containing various columns with information about each object in the cluster. This includes the positions in the equatorial coordinate system (right ascension and declination) and the apparent magnitudes in the U, B, V, R and I bands. `FancyCluster` only uses the B, V and R magnitudes, which correspond to the RGB colours when generating the image.

Not all stars contain all 3 magnitudes in the SIMBAD datasets. When using `FancyCluster`, the user is therefore asked whether they wish to plot only the stars having all the magnitudes or to replace the missing magnitudes with a rough estimation based on the cluster's mean colours.

The colours are estimated in the following way: we calculate the mean colours V-B, R-V and R-B of the cluster. In the case in which only one magnitude is available, the other 2 magnitudes are estimated based on the mean colours. For example, if only the V-magnitude is known, we have:

$$\mathrm{B} = \mathrm{V} - (\mathrm{V} - \mathrm{B})_{\mathrm{mean}}$$

$$\mathrm{R} = \mathrm{V} + (\mathrm{R} - \mathrm{V})_{\mathrm{mean}}$$

If two out of the three magnitudes are known, there are two mean colours which can be used to find the missing magnitude. In that case, we use the average of the two results.
Stars that have none of the 3 magnitudes are always ignored.

The positions are converted from the equatorial system to coordinates centred at the mean centre of the cluster with one coordinate unit corresponding to 1000 arcsec.

Fig. 1 shows an example of a cluster plotted using data from SIMBAD with and without simulated magnitudes for the stars which do not contain all magnitudes in the data.

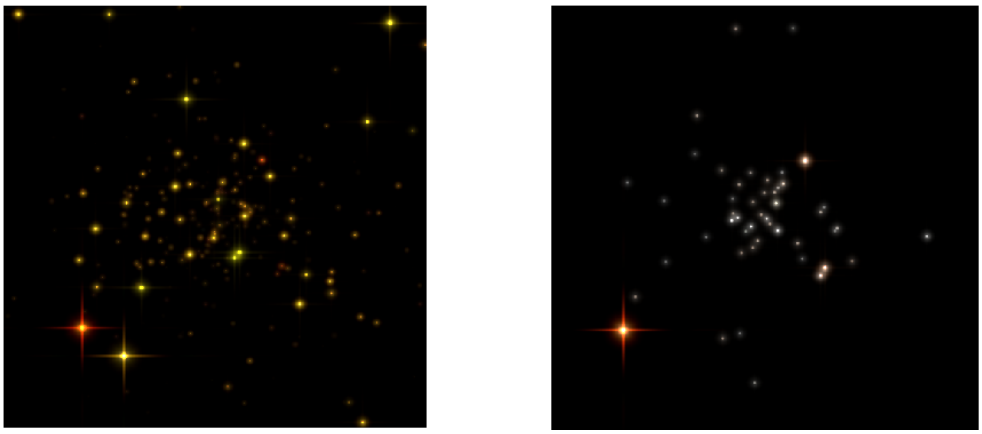

Figure 1: Simulation of M4 cluster using SIMBAD data. Left: including simulated magnitudes missing in the input file. Right: including only stars containing all magnitudes.

3 Frame Settings

The program takes as input the x and y positions of the stars which are placed on the frame according to the system of coordinates set by the user. The input parameters for setting this system are the x and y coordinates of the centre of the image, X_c and Y_c, the size of the plot, Rplot, and the size of the image in pixels, Npix.

The image frame is set such that the central pixel of the frame corresponds to coordinate position (X_c, Y_c) and the frame extends from X_c–Rplot to X_c+Rplot on the x-axis and from $Y_c - \mathrm{Rplot}$ to $Y_c + \mathrm{Rplot}$ on the y-axis. The frame then contains Npix^2 pixels where each pixel position corresponds to a coordinate position following:

$$c_x = \frac{2 \cdot \mathrm{Rplot}}{\mathrm{Npix}} i_x + X_c - \mathrm{Rplot} \qquad (1)$$

$$c_y = \frac{2 \cdot \mathrm{Rplot}}{\mathrm{Npix}} i_y + Y_c - \mathrm{Rplot} \tag{2}$$

c_x = coordinate position on x-axis
c_y = coordinate position on y-axis
i_x = pixel index on x-axis
i_y = pixel index on y-axis

Figure 2 illustrates the correspondence between the system of coordinates and the pixel indices of the frame.

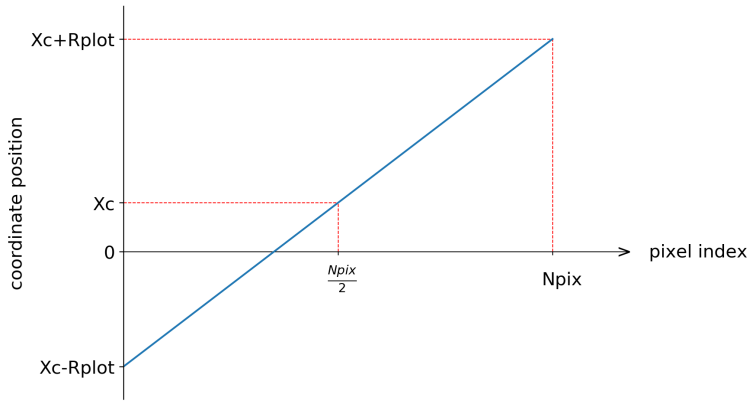

Figure 2: Relation between the positions in the system of coordinates set by the user and the pixel index of the frame where the cluster image is shown, for the x-axis. The y-axis follows the same relation with the corresponding central position.

When reading the input file, the program counts how many stars are in view in the frame according to their positions and informs the user once the simulation is done.

4 Colours

The colours are obtained by approximating the stars as blackbodies. They are calculated based on the luminosity and the temperature of each star supplied by the input file.

The blackbody spectral radiance is calculated using Planck's law at the given temperature T for the wavelength λ:

$$\varepsilon(T, \lambda) = \frac{\frac{2hc^2}{\lambda^5}}{e^{hc/K_B T \lambda} - 1}. \tag{3}$$

Here, h is Planck's constant, c is the speed of light and K_B is the Boltzmann constant. We calculate the spectral radiance in three different wavelengths for

each star, corresponding to the B-, V-, and R-bands. We assign a wavelength of 4500 Å to the B-band, 5600 Å to the V-band, and 6700 Å to the R-band.

The luminosity L_V in the V-band is then found from the ratio between the luminosity and the bolometric correction BC corresponding to this band:

$$L_V = \frac{L}{BC(T, \lambda_{min}, \lambda_{max})}, \tag{4}$$

where the bolometric correction is given by:

$$BC(T, \lambda_{min}, \lambda_{max}) = \frac{\sigma T^4}{\int_{\lambda_{min}}^{\lambda_{max}} \varepsilon(T, \lambda) d\lambda}. \tag{5}$$

Here the integration limits λ_{min} and λ_{max} are the boundaries of the V-band, which we approximate as the central wavelengths of the other two bands.

The magnitude of the V band (M_V) is then calculated using the above-obtained luminosity by the well-known relation:

$$M = -2.5 \log_{10} \left(\frac{L}{L_\odot} \right), \tag{6}$$

with M the absolute magnitude in a particular band and $L_\odot$ the solar luminosity in that band.

Using Eq. 6 for the other bands we get the colours B-V and V-R:

$$B - V = M_B - M_V = -2.5 \log_{10} \left(\frac{L_B}{L_V} \right) \approx -2.5 \log_{10} \left(\frac{\varepsilon(T, \lambda_B)}{\varepsilon(T, \lambda_V)} \right). \tag{7}$$

Similarly,

$$V - R = M_V - M_R \approx -2.5 \log_{10} \left(\frac{\varepsilon(T, \lambda_V)}{\varepsilon(T, \lambda_R)} \right). \tag{8}$$

Using these colours and M_V we obtain the other magnitudes as follows:

$$M_B = M_V + (B - V), \tag{9}$$

$$M_R = (V - R) - M_V. \tag{10}$$

Inserting the absolute magnitudes for bands B and R in Eq. 6 and solving for L we obtain all the required luminosities of the stars.

We created three separate initial frames corresponding to the B, V, and R bands and simulated the stars in each band individually. For generating the final image, we used the `imshow` function from the `numpy` Python package [1], which takes as argument the frames corresponding to the R, G, and B colours. By assigning our bands to the image colours as $(B, V, R) \rightarrow (B, G, R)$, the final image is generated.

4.1 Rescaling the Luminosities

In order to control the luminosity distribution of the cluster we added two rescaling parameters. These two parameters transform the luminosity as:

$$\mathrm{L_{rescaled}} = \mathrm{L} \cdot \mathrm{P1} + \mathrm{P2}. \tag{11}$$

The first parameter, P1, controls the difference in luminosity between the brightest stars and the faintest stars. The smaller it is, the lower the variation of luminosity in the cluster. The second parameter, P2, increases or decreases the total luminosity of the cluster, simply by adding its value to all luminosities. By changing these parameters it is possible to avoid an image where only the brightest stars are visible.

5 Modelling the Stars

Diffraction in the telescope mirrors causes spikes to be seen around bright stars. For this simulation we modeled two types of diffraction spikes, which give the shape of the stars in the image.

The spikes of the first kind (5.1) are bigger and easily recognizable. They can be caused by the edges between the telescope mirrors and by the supporting vanes holding the secondary mirror, such that the number of spikes changes according to the number of edges the mirrors have and the number of vanes. They can be rotated according to the angle of the telescope during the observation.

The second type of spikes (5.2) are smaller, not always visible, and are caused by other effects such as dirty optics. In practice, these look like a large number of smaller spikes surrounding the entire star distributed with varying intensities. At lower resolution or for fainter stars, the small spikes together look like a glow around the star.

Fig. 3 shows an example of both kinds of spikes seen in a picture by the Hubble Space Telescope.

When looking at telescope pictures, we observe big and small spikes only around bright stars while small stars have only a glow around them. Very faint stars appear as simple dots with no glow and no spikes. With this consideration, we made both the big and small spikes appear clearly on the bright stars while making small spikes dominate on fainter stars, with big spikes almost invisible. We then added an extra dot to the center of the star to make very faint stars appear as dots and to make the center of all stars more defined. Its intensity is described by a radial Gaussian function.

We chose two functions to determine the intensities of big and small spikes according to the luminosity L of the host star. These functions are:

$$\mathrm{F_s}(\mathrm{L}) = \mathrm{L}, \tag{12}$$

for the intensity of the small spikes, and

$$\mathrm{F_b}(\mathrm{L}) = \frac{\mathrm{L}^3}{\mathrm{L}^2 + \mathrm{C}}, \tag{13}$$

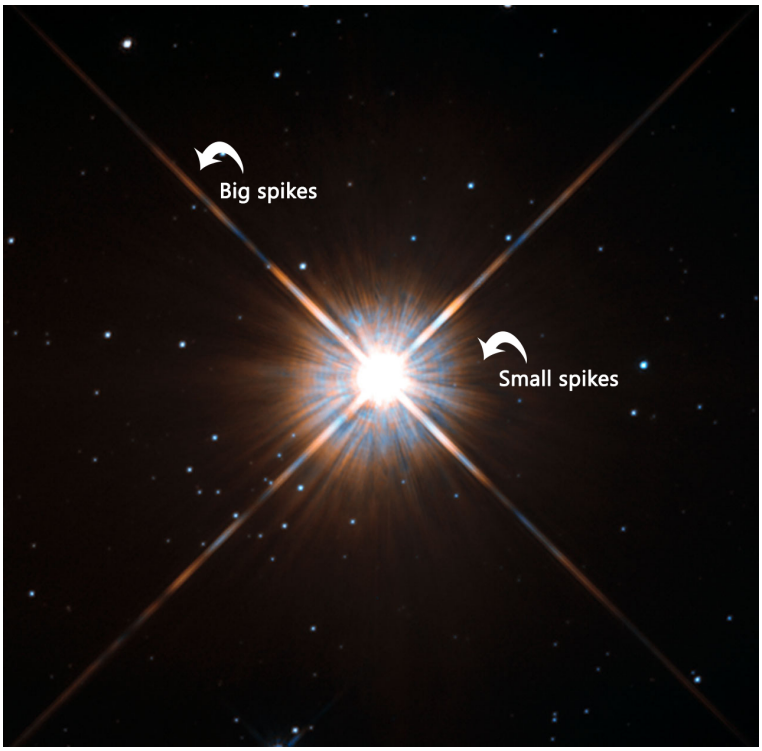

Figure 3: Big and small spikes indicated on a picture of a star from the Hubble Space Telescope Credit: ESA/Hubble and NASA

for the intensity of the big spikes.

This way, the intensities of both spikes go to zero as the luminosity decreases, but the intensity of the big spikes decreases faster with decreasing luminosity. As the luminosity increases, the intensity of the big spikes converges to the intensity of the small spikes, because F_s is an asymptote of F_b. This is illustrated in Fig. 4. Increasing the constant C increases the difference between F_b and F_s for the same luminosity.

For the Gaussian function added to the centre of the star, both the width of the Gaussian and its intensity scale linearly with the luminosity of the star. The width also scales with the PSF, which is a parameter that can be set by the user. This Gaussian is thus given by:

$$I(d) = A \cdot \exp\left(-\frac{(d-\mu)^2}{2\sigma^2}\right), \tag{14}$$

where d is the coordinate distance to the center of the star, μ is 0 as the peak of the Gaussian is centered on the star and σ is the width, given by

$$\sigma = PSF \cdot L, \tag{15}$$

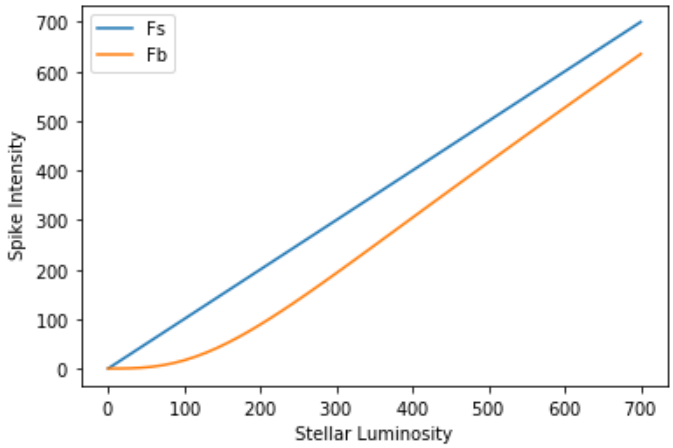

Figure 4: F_s and F_b as a function of the luminosity.

and

$$A = 1.5 \cdot L. \tag{16}$$

Both kinds of spikes have a radial colour pattern which can be described by the Fraunhofer diffraction equation [2]. For modeling this effect we used the equation for diffraction due to light going through a slit and being projected on a surface, which is analogous to the diffraction pattern we observe in the telescope. The intensity I at a distance y from the center of the projection (in our case the distance to the centre of the star) is given by:

$$I = I_0 \frac{\sin^2\left[\frac{\pi a y}{\lambda D}\right]}{\left[\frac{\pi a y}{\lambda D}\right]^2}, \tag{17}$$

where I_0 is the central intensity of the projection, a is the slit size and D is the distance between the slit and the surface where the light is projected.

This equation depends on the wavelength λ and therefore causes the colours to separate radially around the star. The diffraction pattern is simulated for each star and each kind of spike. We multiplied this diffraction pattern by the patterns giving the shapes of the big and small spikes to obtain the desired effect. The value of I_0 is arbitrary since the entire final pattern is later normalized such that the central intensity corresponds to either F_s or F_b depending on the kind of spike.

Finally, the big spikes, small spikes, and the Gaussian dot are all added to the frame at the position of the star.

5.1 Big Diffraction Spikes

For simulating the big spikes we chose to use a cross-shaped spike pattern, which is the pattern seen on images from the Hubble Space Telescope. For this, we create two perpendicular elongated shapes and combine them. The spike rotation angle was added as a parameter that can be set by the user. Figure 5 shows an example of a cluster with big spikes at different angles.

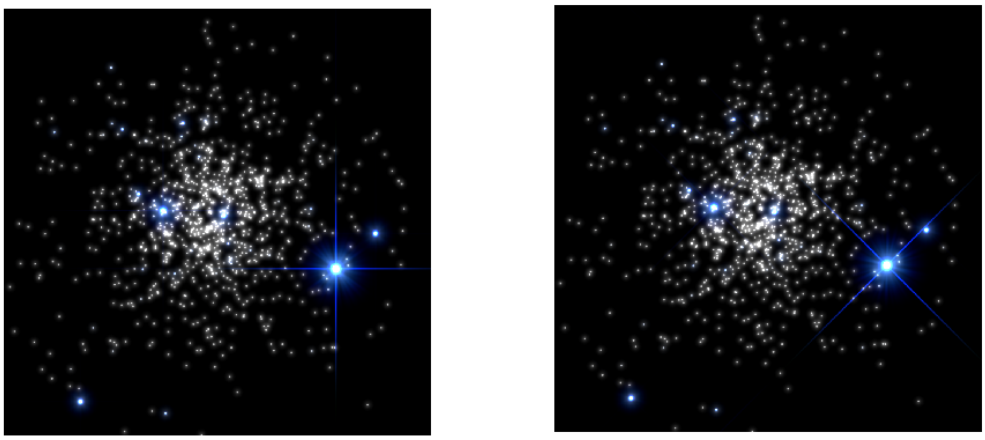

Figure 5: Simulation of a cluster using a standard input file with spikes at different angles. Left: spikes at 0 degrees. Right: spikes at 45 degrees.

The spike intensity function describing the elongated shape consists of a straight line containing the maximum intensity and a Gaussian decay of the intensity as a function of the distance to the line. We chose to use a Gaussian function since the spike is the spread of the light due to the diffraction of all points containing the star, and the image of the star is determined by the point spread function (PSF) which is modeled with a Gaussian. This Gaussian can thus be described by Eq. 14 with d the distance to the line, $\mu = 0$, and σ the PSF. This model ensures that the spike is not one hard line but instead a softer, slightly triangular shape which can also vary in thickness according to the luminosity of the star.

It is important to mention that the PSF used in this case is not the same as the PSF used to create the central dot of the star. We chose to use different values due to graphical reasons, as we judged this choice to create a more realistic image. There are most likely other optical effects involved in the observed shape of the star which we did not take into account and for which we are compensating by adding extra elements to the picture. The PSF determining the thickness of the spikes has a fixed value of 0.002 coordinate units and was not added as a parameter that can be changed by the user in order to avoid confusion. The spike rotation angle was added as a parameter that can be set by the user.

Finally, the two big spikes are summed together, forming the cross-shape,

multiplied by the diffraction pattern, and normalized such that the maximum luminosity is given by F_b. They are then added to the frame.

5.2 Small Diffraction Spikes

The small spikes were created in a similar way to the big spikes: we used the same function describing a line on the frame with maximum luminosity for each spike, but generated 180 of them surrounding the star, at equal angular intervals. This is only an approximation of the pattern we observe in telescope pictures, which seems to be more complex. To create a variation in the intensity of each spike, the intensity was determined randomly for each of them. For each spike, its intensity is a random number between 0 and the luminosity of the star. This pattern is the same for every star in the picture and changes every time the program is executed.

To ensure that the big spikes are indeed longer than the small spikes, we multiplied the small spikes' pattern by another radial Gaussian function centered on the star. The width of the Gaussian, therefore, limits the extent of the small spikes and is given by $0.001 \cdot L$.

The spike pattern is then multiplied by the diffraction pattern and by the extra Gaussian function used to make them shorter. Lastly, the small spikes are normalized such that the central intensity is given by F_s.

Fig. 6 shows an example of a single simulated star with both kinds of spikes.

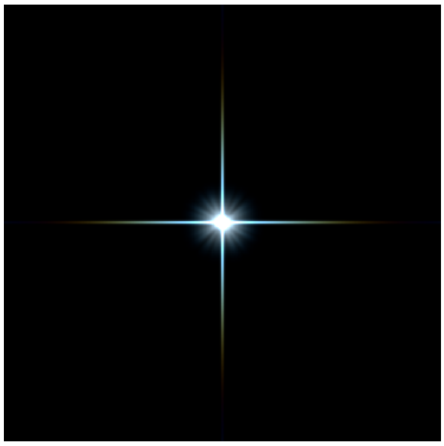

Figure 6: Simulation of a single star having small and big spikes.

6 Discussion

Our algorithm creates realistic visual representations of stellar clusters and can be used for simulating predicted arrangements of the cluster and for displaying the current stars. However, many improvements can be made to future versions of this code.

The most computationally expensive part of the code is the simulation of the spikes. This has been greatly reduced by generating the pattern for small spikes before starting the iterations for each star and applying the same pattern to all stars. The big spikes, however, are still created at each iteration. By using the same strategy for them as was used for the small spikes, the computation time can be further reduced.

When considering the goal of creating a realistic picture of the cluster, some of the visual components that create the shape of the stars are not based on any physical laws and were simply added to improve the similarity to telescope pictures. These would be the pattern of the small spikes, the law for setting their brightness with respect to the big spikes, and the extra Gaussian added to the center. This can be improved with further study on telescope optics for a more detailed and accurate simulation of each effect.

Another issue to be addressed is the estimation of the colours of the stars missing from SIMBAD data. Right now we use an extremely rough estimation of them with the sole purpose of making visualization of their positions possible. A much more advanced method should be used for a realistic estimation of their colours based on physical or empirical laws.

Given that a main use for this package would be the making of simulations of the future of the cluster, an interesting addition would be a feature for creating animations of the stellar positions in time. For this reason, the input file already contains the columns for the time and velocities of each star. In a future version of `FancyCluster` this could be implemented, also taking the position along the z-axis into account.

7 Conclusion

We have developed a Python script that creates visual images from cluster data using either their luminosity and temperatures or the colour magnitudes of the stars. These are modified to look like a telescopic image taking into account visual effects that affect a realistic image.

The shape of the stars is mostly determined by diffraction spikes. We categorized the spikes into two types: big spikes and small spikes. The big spikes are more prominent and easily recognizable, resulting from the edges between telescope mirrors and the supporting vanes holding the secondary mirror. Small spikes are smaller and not always visible, often caused by factors such as dirty optics. The most computationally heavy part in our code was the generation of small spikes which was optimized by creating the same pattern for all stars in the image.

The intensities of the spikes are determined by two functions based on the luminosity of the star, such that the ratio between the intensities of the different kinds of spikes changes according to the brightness of the star. We additionally included an extra Gaussian dot at the center of the star to make it more defined and help recreate the observed phenomena.

The final image is the result of a combination of physical effects and graphical components added for improved visualization. We have added plenty of optional input parameters that can be modified by the user. These parameters can be used to obtain the desired image according to the user's needs and preferences.

8 Acknowledgments

We would like to thank our professor, Dr.Michela Mapelli for her support in providing a structure for the initial version of the code `FancyCluster`. We are also grateful to the SIMBAD database for providing observational data used to test our code.

References

[1] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[2] R Nave. *Fraunhofer Single Slit*. URL: http://hyperphysics.phy-astr.gsu.edu/hbase/phyopt/sinslit.html.

[3] M. Wenger et al. "The SIMBAD astronomical database". In: *Astronomy and Astrophysics Supplement Series* 143.1 (Apr. 2000), pp. 9–22. DOI: 10.1051/aas:2000332. URL: https://doi.org/10.1051.

9 Appendix: User's Guide

For using `FancyCluster`, the function `makeimg` should be used:

import fancycluster
fancycluster.makeimg(filename, simbadfile=False, cpos=(0,0), Rplot=1, angle=0, savefig=True, output_file="fancyfig.png", Npix=500, diff_params=(800,0.1), C=300, rescaling=(0.01,0.3), PSF=0.00003)

Parameters:

filename: ***string***
Name (and path, if needed) of the input file.
simbadfile: ***bool, optional***

Option to use .ascii file from the SIMBAD database as input.

cpos: ***array_like, shape (2,) optional***

Coordinate position (X_c,Y_c) of the centre of the frame.

Rplot: ***float or int, optional***

Dimension of system of coordinates with respect to the pixel index. A bigger value gives a more compact image of the cluster. See Section 3 of the documentation.

angle: ***float or int, optional***

Rotation angle of diffraction spikes in radians. When 0, the spikes are parallel to the edges of the frame.

savefig: ***bool, optional***

Option to save the figure. If True, it plots the figure and saves it with the name given by output_file. If False, it plots the figure without saving it.

output_file: ***str, optional***

Name of the output file to be saved.

Npix: ***float or int, optional***

Size of the image in pixels. The entire image has a total of Npix^2 pixels.

diff_params: ***array_like, shape (2,), optional***

Diffraction parameters in the Fraunhofer equation for diffraction due to a slit, used for creating the diffraction spikes. These are (a,D), with a the slit size and D the distance between the slit and the projection. See Eq. 17 in the documentation.

C: ***float or int, optional***

Parameter for setting the sensitivity of the intensity of the big spikes according to the stellar luminosity. See Eq. 13 in the documentation.

rescaling: ***array_like, shape (2,), optional***

Rescaling parameters (P1,P2) for the luminosities. See Eq. 11 in the documentation.

PSF: ***float or int, optional***

Value of the telescope's PSF, sets the size of the central dot of the star.