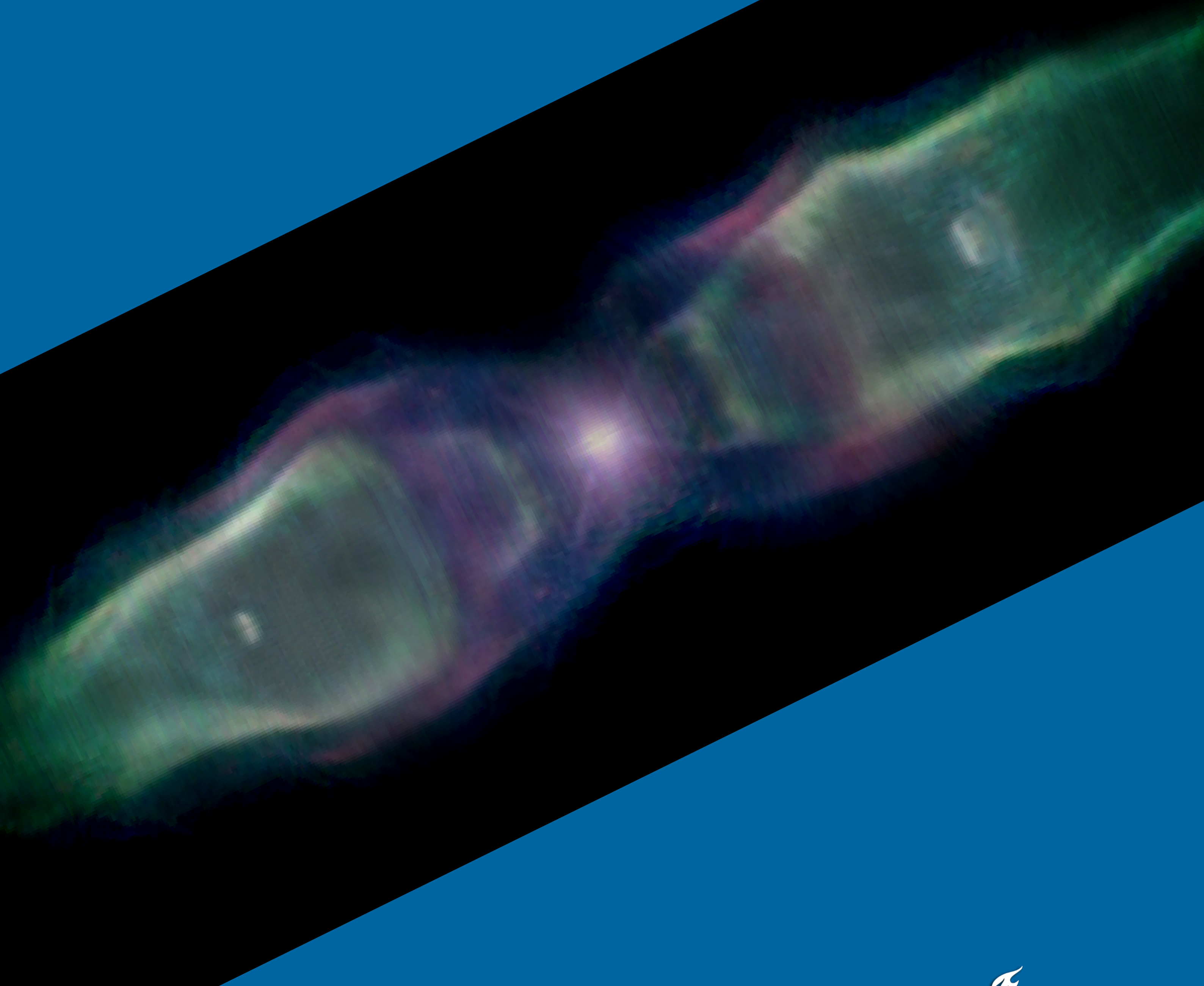
Accelerating Axial-Symmetrical Nebulae Visualization and Reconstruction

Nouri Khalass



Accelerating Axial-Symmetrical Nebulae Visualization and Reconstruction

by

N.M.P. Khalass

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday April 8, 2021 at 10:00 AM.

Student number:	4742397	
Thesis committee:	Prof. dr. E. Eisemann,	TU Delft, supervisor
	Dr. ir. R. Bidarra,	TU Delft
	Dr. E. Isufi,	TU Delft

This thesis is dedicated to the loving memory of my father,
still missed dearly every day and,
without whom this thesis would not be possible

Ahmad Khalass

Abstract

Nebulae are colorful astronomical phenomena that have a mesmerizing appearance. Researching them is difficult, since they are many light-years away. As a result of this, we can only observe them from one viewpoint: Earth. The appearance of nebulae is almost always unique, because of the many factors that influence it. Still, there exist a class of nebulae with a pronounced axial-symmetry in their appearance. The axial-symmetry in these nebulae gives us an indication on how particles in the nebulae will be distributed. We can exploit this symmetry and represent the 3D volume of such a nebulae using a 2D map that is rotated around the symmetry axis. Previous methods converted the 2D map into a 3D volume before rendering, but we propose a novel visualization technique whereby only the 2D map is used. A benefit of representing the nebula using a 2D map is that it is easier to modify than a 3D volume, making it easy to model plausible looking nebulae. We also propose new techniques for reconstructing the 2D map of existing nebulae from telescope images. Combining our achievements in visualization and reconstruction makes it possible to view real axial-symmetrical nebulae from novel viewpoints. Finally, we propose methods to reintroduce asymmetries in volumes of the axial-symmetrical nebulae. These irregularities are also found in real axial-symmetrical nebulae and make the visualizations appear even more realistic.

Acknowledgement

A wise man once said “het komt wel goed”. Believing in this was not always easy, but I tried to hold on to it during the development of my thesis, no matter how difficult it was. But believing it made it true, because I am finally here and I can say that I have completed my master’s thesis.

I wish to thank various people for their contribution to this project. I would like to express my appreciation to Prof. Dr. Elmar Eisemann, my research supervisor, for being open to the idea of doing a thesis on nebulae visualization. I have a huge amount of respect for his knowledge of, enthusiasm towards and dedication to Computer Graphics and Visualization. I also would like to thank my co-supervisor Ir. Annemieke Verbraeck for her extensive support, willingness to help and constructive feedback. Finally, I would like to thank my friends and family for their unconditional support, love, and encouragement throughout this adventure.

Vlaardingen, April 2021

Contents

List of Figures

1 Introduction

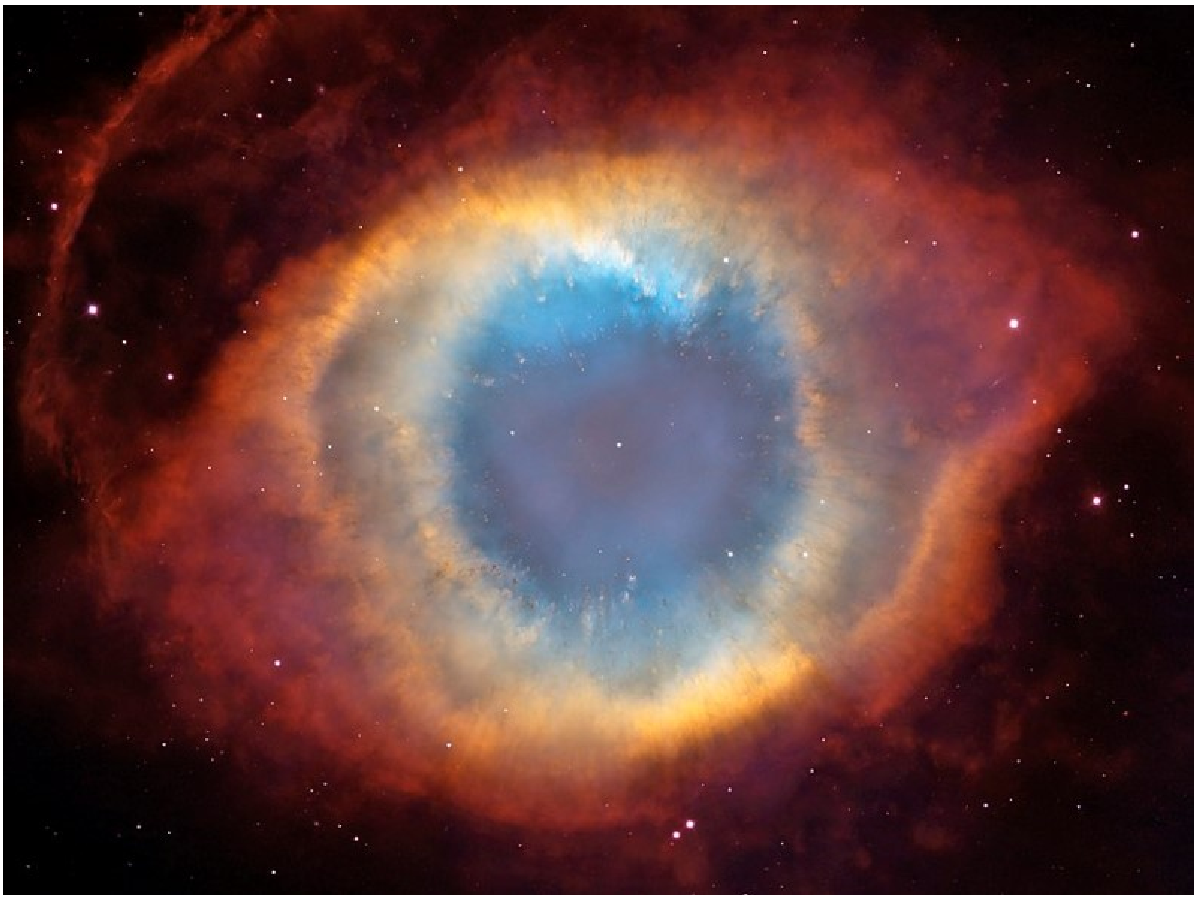


Figure 1.1: Helix Nebula [44]

Nebulae are one of nature's most beautiful sites to behold. Their intricate shape and vibrant colors make them very unique. While most of the known universe is empty and nebulae form only a small group in the large collection of astronomical objects, they have been depicted countless times in film and in video games because they characterise the mystery and, more importantly, the beauty surrounding the unknown universe so well.

The largest hurdle in research on nebula (or any other astronomical objects) is the large distance that they are away from Earth. The nearest nebula is the "Helix Nebula" (Fig. 1.1) and it is approximately 650 light-years away. Because of the large distance between us and nebulae, we cannot visit them and, given the restrictions on resolution, we can only perceive them from one point-of-view. This constraints us in the amount of information that we can gather on them. As a result of this, we must often make assumptions on the structure and composition of nebulae.

Most research on nebulae in the field of Computer Science has been on methods to recover the shape and inner structure of nebulae. If a plausible particle distribution is recovered, it should be possible to view nebulae from novel viewpoints, which, as we said, is normally not possible. This reconstruction problem is ill-posed, so simplifications to the problem must be made in order to come to plausible results. Many works, including ours, use the fact that a particular group of nebulae has a specific property which makes the reconstruction problem easier to solve. By focusing on nebulae with an axial-symmetrical appearance, we have some indication on how particles in these nebulae will be distributed. Previous reconstruction methods give plausible results, but can take several hours to compute, which limits their usability. We address this shortcoming with the advances that we make in axial-symmetrical nebulae visualization.

Research on nebulae visualization is rare, and we propose a new method for visualizing axial-symmetrical nebulae in this thesis. There exist no visualization systems that take into consideration the axial-symmetry that is present in axial-symmetrical nebulae. Modeling an axial-symmetrical nebulae can therefor be a difficult task. We use a simplified representation of axial-symmetrical nebulae which makes it easier to model a nebula.

Our contributions to the field of (nebula) visualization and reconstruction are the following:

1. a novel visualization technique for axial-symmetrical nebulae which uses a memory efficient and easy to manipulate representation for the particle distribution of the nebula,

2. a tool to view and manipulate an axial-symmetrical nebula and with which shape and color deformations can be applied to reintroduce asymmetry in the appearance of the nebula,

3. a novel reconstruction approach which can reconstruct, from a picture, the particle distribution of a nebula with a strong (axial-)symmetrical appearance, such that the nebula can be viewed from novel viewpoints and also be modified using our tool.

With our work, we hope to make it possible to create illustration material suitable for planetariums and for other educational purposes.

In the coming Chapter (Chapter 2), we will give background information that is relevant to the topic of this thesis. Then in Chapter 3, we discuss the workings of our rendering and reconstruction techniques. Chapter 4 will go over how our tool works and how it can be used. In Chapter 5, we will measure how well our rendering method performs and show results of our reconstruction technique. These results will be discussed in Chapter 6 and we will comment on when our methods perform best, as well as its shortcomings. Finally, in Chapter 7, we will conclude this thesis with a summary of everything that has been discussed and bring attention to some points that could be researched in the future.

2 Background

In this Chapter, we will share background information that is relevant for the topic of this thesis. In Section 2.1, we will provide information on the appearance, composition and formation and of various types of nebulae. Section 2.2 will discuss the different methods that have been used thus far to visualize nebula on a computer. Then, in Section 2.3, we go over the methods used to reconstruct the volume of an existing nebula from an image captured with a telescope.

2.1. Nebula

Nebulae are distant astronomical objects. In images of nebulae made with a telescope, they often have a cloud-like appearance. The shape of a nebula is almost always unique. In general, they consist of interstellar dust particles and different types of gas such as hydrogen, nitrogen and helium [21, 23]. They can radiate light at different wavelengths, such as ultraviolet light or infrared light. These are not visible by us humans, but some nebulae also radiate light in the visible spectrum, which allows them to be perceived with the naked eye.

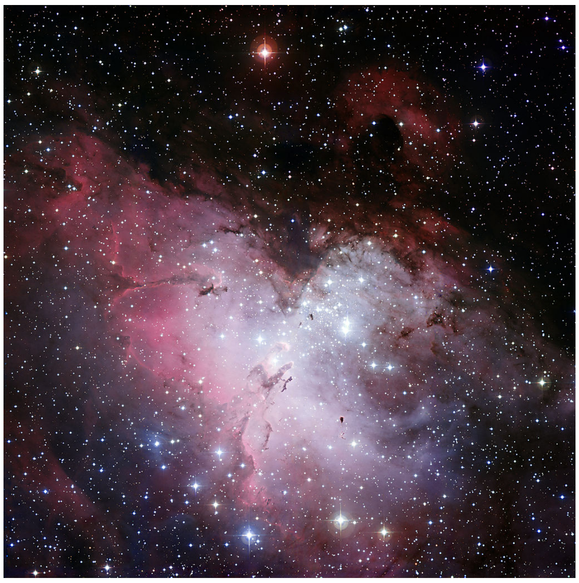


Figure 2.1: Eagle nebula [7]

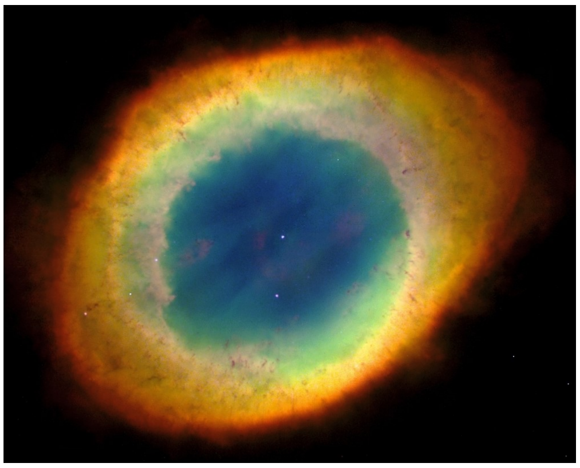


Figure 2.2: M57 nebula [28]

The reason that nebulae vary so widely in their appearance has two primary reasons, each of which will be discussed separately. The first factor that determines the appearance of a nebula is its shape, which is a result of how its particles are distributed in space. We will dive into this in Section 2.1.1. The second factor that determines the appearance of a nebula is how it radiates light. Some nebulae consist of material that emit light while other nebulae only propagate the light emitted by surrounding stars. Some of this light is lost due to absorption and this effect is also visible to us. All of this is discussed in Section 2.1.2.

Nebula that have a strong symmetry in their appearance and where scattering and absorption play no role in the appearance of the nebula are the most relevant for this thesis. This is because these qualities make them very suitable for reconstruction. More on this in Section 2.3.

2.1.1. The different shapes of nebulae

The shape of a nebula can vary quite widely. Some nebulae have a structure that appears completely random (Figure 2.1) while others look less chaotic (Figure 2.2). Nebulae can even have a very symmetrical appearance that is similar to a sphere or a butterfly. We will distinguish two categories for the different shapes that a nebulae can have, *symmetrical* and *asymmetrical*. Note that as nebulae consist of particles and gas in space, nebulae with a symmetric shape will still never be perfectly symmetrical.

Symmetrical nebulae

The most common form of symmetry in nebulae is axial-symmetry, an example is the NGC 7009 nebula in Figure 2.3. Some nebulae are not only symmetric with respect to an axis, but in all directions from the center. An example of a point-symmetric nebula is the Spirograph nebula in Figure 2.4.

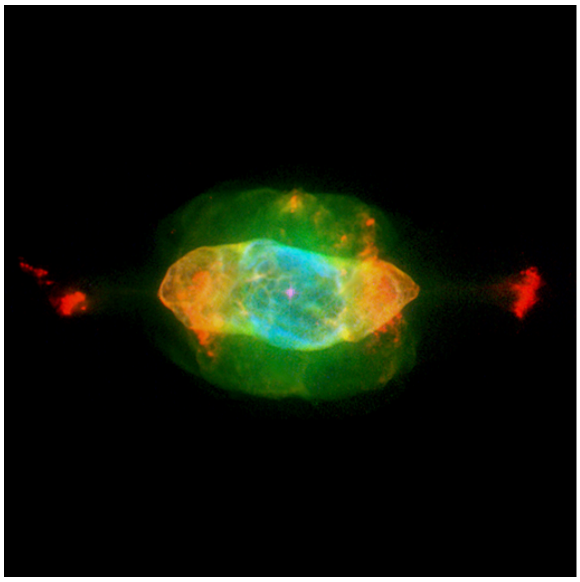


Figure 2.3: NGC 7009 nebula [3]

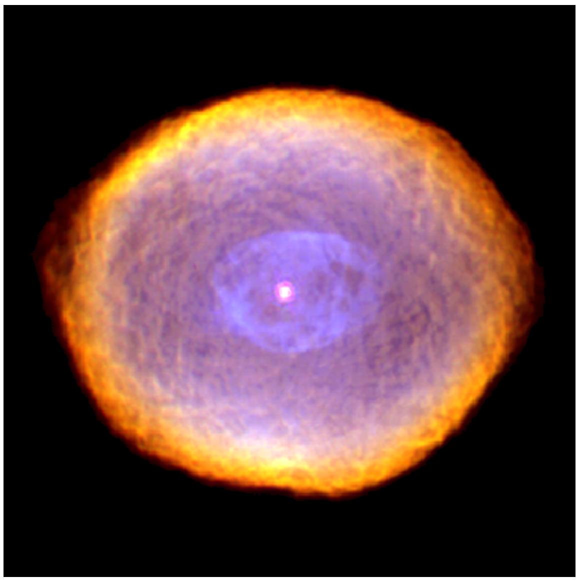


Figure 2.4: Spirograph nebula [34]

Extensive research has been done on why some nebulae can have a symmetrical appearance [18]. This research has led to the *interacting stellar winds theory* [23] which explains why some nebulae have an axis-symmetrical particle distribution. This theory proposes that at the center of these strongly symmetrical nebulae often lie one or several stars. These center stars emit gas throughout their lifetime. This gas accumulates at the equatorial plane of the stars. As the stars enter their final stage they start to emit gas at a much higher velocity. This new gas is hotter and moves faster than the older gas that is accumulated. As a result, the new gas that is emitted is blocked by the old gas and is redirected away from the equator. The gas forms lobes on the two sides of the equator of the stars. This pushing and pulling is known as interstellar winds and not only affects the gas but also interstellar dust. A visualization of the interacting stellar winds theory is shown in Figure 2.5.

The theory explains the shape of axial-symmetrical nebula like in Figure 2.3, and to a lesser extend the shape of nebulae that are point-symmetrical such as in Figure 2.4, as their particles are pushed much more uniformly in all directions.

Asymmetrical nebulae

While the shape of a symmetrical nebula can often be explained using the interacting winds theory, the asymmetry in asymmetrical nebulae has a variety of causes. Some of the asymmetrical nebulae are remnants of supernovae, the event that occurs when a star has reached the end of its lifespan. During this explosive event,

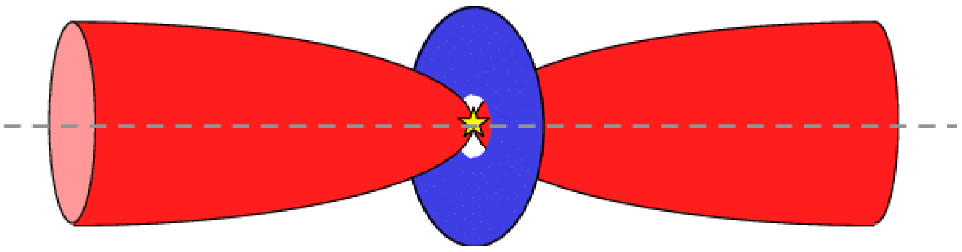


Figure 2.5: Visual explanation of the interacting stellar winds theory which explains the shape of (axial-)symmetrical nebula. The equatorial plane of the center star is shown in blue. Gas is accumulated in the region of this plane throughout the lifetime of the star. In the final stage of the star's life, hotter and faster gas is emitted which clashes with gas that has accumulated around the equatorial plane. As a result, the gas is pushed away from the equatorial plane in the regions marked in red. Figure from Magnor et al. [23]

interstellar material is ejected outwards from the star with high force and scattered chaotically around the dying star. Other asymmetrical nebulae thank their shape from stars within their interstellar dust cloud, that push around the interstellar material non-uniformly.

Most of the nebulae that are known to us are asymmetrical nebula, examples are Figure 2.6 and Figure 2.7.

Figure 2.6: Orion nebula [33]

Figure 2.7: Witch Head nebula [35]

2.1.2. The behaviour of light in nebulae

Light that comes from a nebula can travel in a variety of ways before it reaches us. A nebula can contain one or several stars which emit light, but often this light does not reach us directly, as it is scattered between interstellar dust particles before it leaves the nebula. Some of the light is absorbed by interstellar dust particles and therefor does not reach us. Nebulae where reflection and absorption play a significant role in the appearance of the nebula are called **reflection nebulae**. Examples of reflection nebulae are the LDN 1251 nebula (Figure 2.8) and the NGC 2261 nebula (Figure 2.9).

In addition to reflecting light from stars inside the nebula, nebulae can also emit light themselves. When most of the light that a nebula emits comes directly from its interstellar dust then we call such a nebula an **emission nebula**. The interstellar dust in an emission nebula is extremely hot. This is because it is ionized by radiation coming from one or several stars within the nebula. It becomes so hot that it starts to radiate its own light. The Orion nebula (Figure 2.6) and the M57 nebula (Figure 2.2) are examples of emissive nebula.

In this thesis we primarily focus on nebulae that are both symmetrical and purely emissive, as these properties make them suitable for algorithms that aim to reconstruct the 3D shape of the nebulae. More on the reconstruction in Section 2.3.

Conveniently, many symmetrical nebulae are also emission nebulae. The interacting stellar winds theory

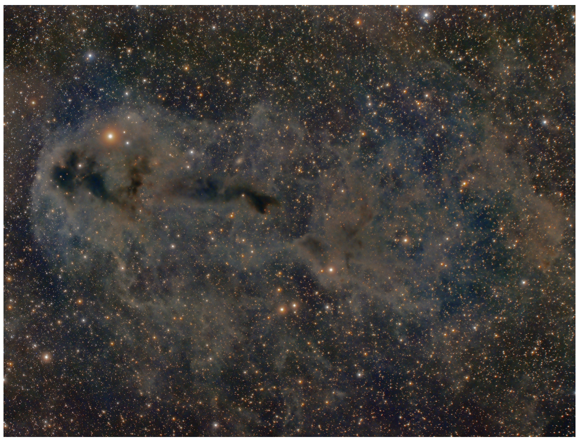


Figure 2.8: LDN 1251 nebula. Notice the dark spots where all light coming from stars behind the nebula is absorbed [36].

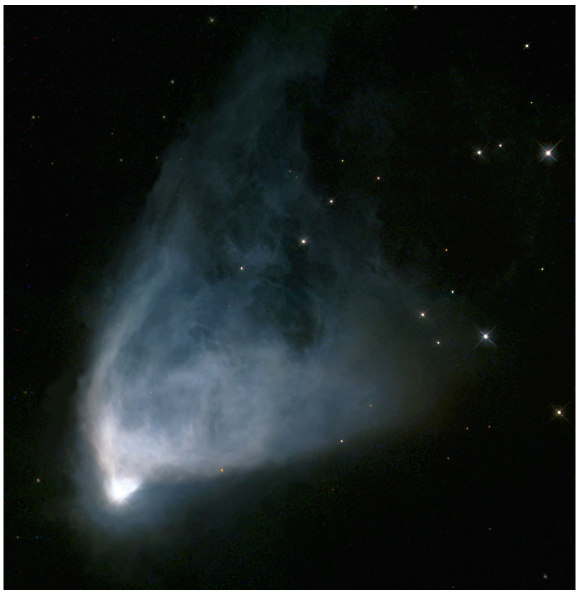


Figure 2.9: NGC 2261 nebula. Notice how a star illuminates the interstellar dust whereby radiance is scattered throughout the nebula [27].

explains the symmetry, but also the emissive character of these nebulae. The winds pushes away the large dust particles and molecules which would lead to scattering and absorption. The remaining interstellar material is distributed symmetrically throughout the nebula and ionized by the dying star. Symmetrical emission nebulae are often called **planetary nebulae**. This name might imply that these nebulae have anything to do with actual planets, but this is not the case. Examples of planetary nebulae the NGC 6826 nebula (Figure 2.10) and the NGC 6302 nebula (Figure 2.11).

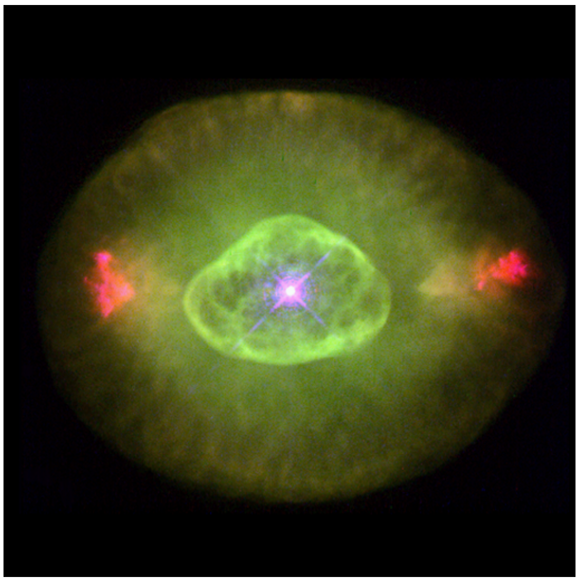


Figure 2.10: NGC 6826 nebula [2]

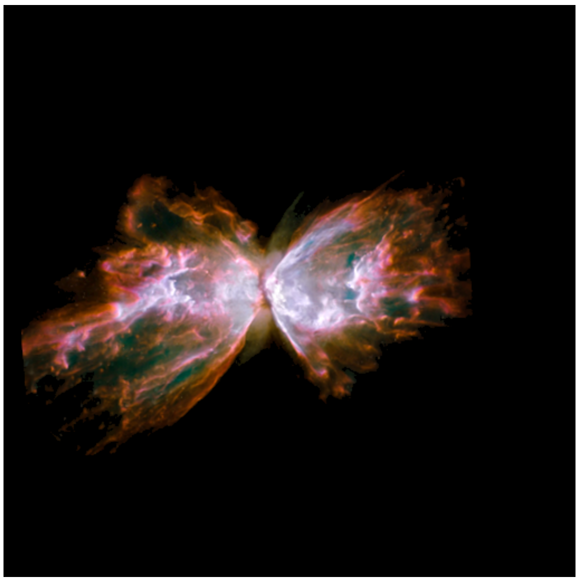


Figure 2.11: NGC 6302 nebula [29]

2.2. Nebula Visualization

In the field of astronomy, Computer Graphics has been used extensively to visualize many phenomena [24]. Examples are the work of Verbraeck and Eisemann [46] on visualizing the effect of black holes on space curvature in real-time and the research of Hildebrand et al. [12] on the reconstruction and visualization of spiral galaxies. One of the earliest works of nebula visualization is by Nadeau et al. [26] on visualizing the Orion nebula, which is one of the closest nebulae to earth.

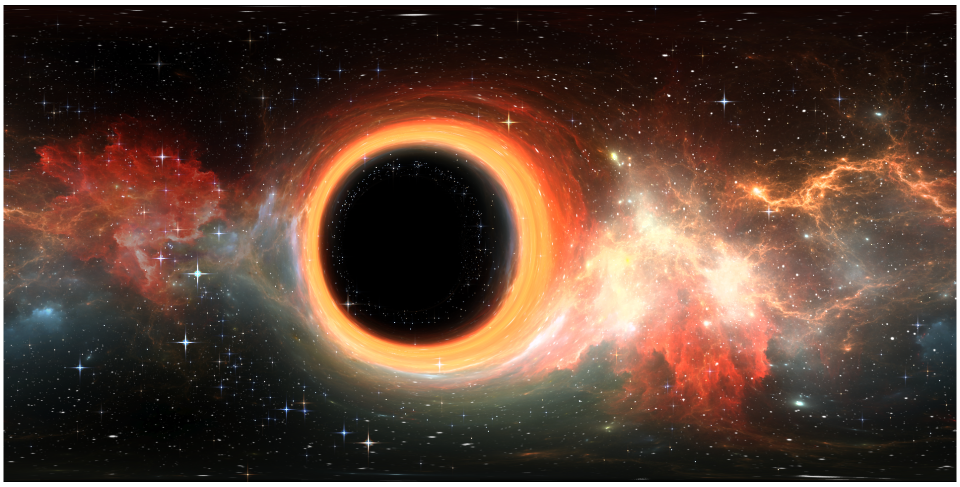


Figure 2.12: Visualizing black holes by Verbraeck and Eisemann. [46]

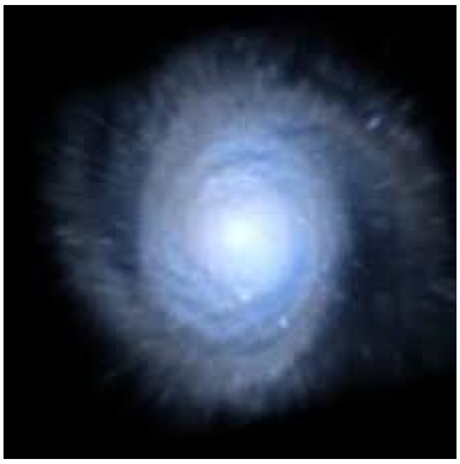


Figure 2.13: Rendition of a Spiral Galaxy by Hildebrand et al. [12]

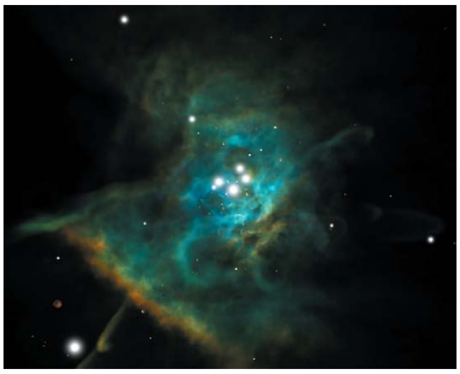


Figure 2.14: Visualization of the Orion nebula by Nadeau et al. [26]

Figure 2.15: Examples of using Computer Graphics to visualize astronomical phenomena

Most of the research on nebula visualization has focused on visualizing asymmetrical reflection nebula. Magnor et al. [22] created an interactive visualization tool for realistically rendering the 3D dust distribution illuminated by nearby stars. The work of Gislason et al. [9]) is on real-time visualization of radiative transfer in reflection nebula. Steffen et al. [45] have created the tool *Shape* that can be used as a 3D modeling tool for astronomical phenomena. As mentioned in the previous Section, the appearance of reflection nebula is heavily influenced by the scattering of light happening inside the volume. For this reason, the rendering systems need to compute relatively complex radiate transfer functions in order to have a realistic result. On top of that, the nebulae are often completely asymmetrical, which means that no optimizations based on symmetry are possible.

A simpler rendering model is used by the aforementioned Nadeau et al. [26] to visualize the Orion nebula. It only takes into consideration emission and absorption. Even with this simplification, they are able to model the ionized gasses and dense dust clouds of the Orion nebula which makes their reproduction rela-

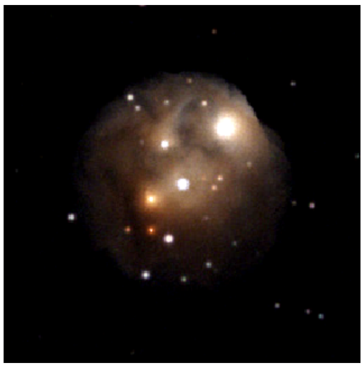


Figure 2.16: Rendition by Magnor et al. [22] of the “Cocoon” nebula

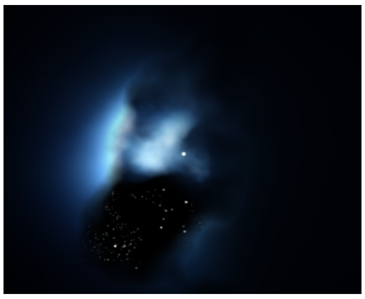


Figure 2.17: Visualization of radiative transfer in reflection nebula by Gislason et al. [9]

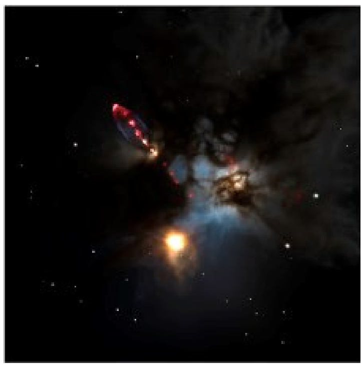


Figure 2.18: Nebula created with Shape by Steffen et al. [45]

Figure 2.19: Computer visualizations of reflection nebula

tively faithful.

To our knowledge, there exist no rendering systems specifically for rendering axial-symmetrical nebulae (or axial-symmetrical volumes in general), where a simplified representation of the volume is used, instead of a complete 3D model. Creating a rendering system specifically for axial-symmetrical nebulae is a goal of this Thesis. Like the rendering system by Nadeau et al, our rendering system will only take into consideration emission and absorption. The effect of light scattering, whether that light comes from ionized material or from stars, will not be taken into consideration.

In Chapter 3, we will explain how our rendering system works, but first we explain how a conventional volume rendering system works. With “conventional” we primarily mean that the way in which the volume is represented is with a 3D volume description and that the axial-symmetry in the volume is not taken into consideration.

2.2.1. Volume Rendering

Representing a nebula as a volume can be done using voxels. These voxels sit in a cuboidal 3D grid. Each voxel has one or more properties that characterize the region of space that the voxel corresponds with. Each voxel stores its emission and absorption values as well as the color of the light that it emits.

This 3D grid of voxels is a 3D volumetric representation of the nebula. Visualizing this 3D volume is achieved by projecting it onto a 2D image plane. Because nebulae are so far away, we can assume that all light rays coming from a nebula are parallel to each other. A telescope looking at a nebula in space, captures these parallel light rays on a sensor, making an image. To visualize our virtual volume, we ask ourselves what the telescope would see if it was looking at the nebula from a certain angle. Instead of capturing light rays, we send rays from our 2D image plane into the virtual volume to find the light contributing to each ray. Like a real telescope sensor, the image plane consists of pixels, but instead of each sensor ’pixel’ receiving light rays, we send rays from the pixels of this image plane.

The origin O of the ray is the position of the pixel that the ray comes from in the image plane in the 3D scene. The direction D of the ray is a unit vector perpendicular to the image plane. With O and D we can describe a ray going through 3D space that starts at O and has direction D. A point P_t along this ray has 3 components: x, y and z. We denote the x component of P_t as x_{P_t}, the z component of D as z_D, etc. The equation for a ray is shown in Equation 2.1.

$$P_t := O + t \cdot D \tag{2.1}$$

$$= \begin{pmatrix} x_O + t \cdot x_D \\ y_O + t \cdot y_D \\ z_O + t \cdot z_D \end{pmatrix} \tag{2.2}$$

To know whether a ray intersects with the volume, we check whether it intersects with the bounding cube that encompasses the whole volume. If there is an intersection then there is a point where the ray enters the cube ($P_{t_{entry}}$) and where it exits the cube ($P_{t_{exit}}$), (for simplicity P_{entry} and P_{exit}). Computing the points of intersection of a ray and a cube has been researched extensively [6, 14, 25].

If a ray does not intersect with the bounding cube then no radiance is received by the corresponding pixel in the image plane (perhaps only a background color). When a ray does intersect with the bounding cube, we want to accumulate all radiance that lies between P_{entry} and P_{exit}. This is the radiance that is received by the pixel in the image plane that the ray corresponds with. This, along with all other concepts discussed so far, is shown schematically in Figure 2.20.

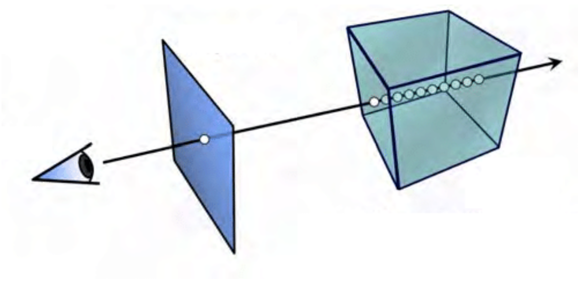


Figure 2.20: Ray traveling from the camera through the image plane into the volume.

Each voxel that is traversed by the ray can influence the accumulated radiance that is captured. Since in this thesis we are dealing with a volume that is only emissive, there is no scattering. In other words, voxels that the ray does not traverse do not influence the accumulated radiance of the pixel. Each voxel has a certain emission value a_i, extinction value b_i and color c_i. The amount of light emission of a voxel is represented by the emission value. The extinction value represents the amount that a voxel attenuates the radiance along the ray, a high extinction in one voxel reduces the influence of the following voxels. The accumulated radiance C that is received by a pixel in the image plane whose ray corresponds with P_{entry} and P_{exit} can be computed with Equation 2.3.

$$C = \int_{P_{entry}}^{P_{exit}} a_v * c_v \cdot e^{-\int_0^v b_t dt} dv \qquad \text{(Continuous form)} \qquad (2.3)$$

$$C = \sum_{i=0}^{n} c_i * a_i \prod_{j=0}^{i-1} (1 - b_j) \qquad \text{(Discrete form I)} \qquad (2.4)$$

$$C = c_0 * a_0 + (1 - b_0) \cdot (c_1 * a_1 + (...)) \qquad \text{(Discrete form II)} \qquad (2.5)$$

Equation 2.3 is a continuous formula for sampling the ray through the volume. However, a discrete version (Equation 2.4) can be solved much easier and converges towards the actual result of the integral when more and more points on the ray are sampled. For this discrete version the volume is sampled at constant intervals along the ray. This interval value is equal for all rays.

Equation 2.5 shows more clearly that the $(1 - a_i)$ term determines how much the point $i + 1$ contributes to the output color. If point i is fully opaque ($a_i = 1$) then the point(s) after i will not contribute anything to the output color. Note that in some systems the a_i term is already incorporated in the c_i term and thus not denoted explicitly.

We can summarize all of this in Algorithm 1 which provides the pseudo-code of a volume-rendering implementation.

Algorithm 1 Pseudo-code implementation of Volume Rendering

```
Require: $O, D \leftarrow$ Origin and direction of ray
Require: $t_{entry}, t_{exit} \leftarrow$ Entry point and exit point of ray with bounding cube
  $C_{acc} \leftarrow 0$                          ▷ Accumulated radiance
  $A_{acc} \leftarrow 0$                          ▷ Accumulated extinction
  for $t \leftarrow t_{entry}, t_{exit}$ do           ▷ Step from the entry point to the exit point
      $P \leftarrow O + t \cdot D$                ▷ Compute current position in bounding cube
      $c, a, b \leftarrow V[P]$                   ▷ Retrieve color, emission and extinction values
      $C_{acc} \leftarrow C_{acc} + (1 - A_{acc}) \cdot (a * c)$
      $A_{acc} \leftarrow A_{acc} + b$
  end for
```

2.2.2. Accelerating Volume Rendering

Many different techniques exist to accelerate the performance of Volume Rendering. We are going to discuss two approaches which will be relevant when we discuss our own acceleration techniques. The two techniques approach the problem in completely separate ways, but they can be used together with each other.

GPU Acceleration As explained in the previous Section, the volume-rendering algorithm works by tracing a ray, coming from a pixel on the image plane, through the volume and sampling intervals of the ray. The radiance that is received by one pixel is not influenced by the radiance that is received by other pixels. This means that computing the radiance that the pixels receive can be done in parallel. Since there are often a large number of pixels, this leads to a significant speedup. This parallel workload is very suited for the GPU, which can perform large amounts of computation in parallel [1]. Many programming frameworks such as CUDA [38] and OpenCL [10] exist to help with GPU programming.

For us, the CUDA [38] framework is relevant because we make use of it. CUDA is an often used GPU programming framework from NVIDIA for their GPUs. To harness the massive amount of computation power that lies in the GPU, the CUDA programming model allows access to the GPU with “kernel programs”. These kernel programs are executed by threads and a set of threads belong to a certain thread block. These thread blocks are executed in parallel. Each thread has a unique index and also knows the block index of the block it belongs to. With all this information, each thread can be uniquely identified and can know, for example, which pixel in the image plane it is evaluating.

A kernel program can write to memory that is shared between threads. The fact that threads can access memory at the same time can lead to race-conditions and stalls. All of these are highly undesirable in regards to performance. The memory that a kernel has access to is stored on the GPU. Copying data from and to the GPU can be an expensive operation, but is sometimes necessary to, for example, store results to disk.

Octrees To exploit empty space (or homogeneous space) that can be present in a volume, an octree can be used to store information about multiple voxels at once, rather then storing each individual voxel value. The use of octrees reduces the amount of memory necessary to store the volume and can improve the performance of Volume Rendering [17, 43].

The root of the octree represents the entire volume. The volume is recursively subdivided into eight equally-sized regions until a certain level governed by a strategy. This strategy could be that a region is subdivided when it is not completely empty (and can still be subdivided). Another approach is to subdivide a region when the contents of that region is uniform (but not necessarily empty).

Because information about homogeneous regions does not have to be stored per voxel, using an octree can be used to decrease the amount of memory that is used to store the volume. In addition to that, if a segment of a ray is known to fall in a homogeneous region of space, techniques can be employed to quickly integrate over that section of the ray, instead of sampling at intervals, which accelerates the volume rendering.

Figure 2.21 shows how an octree can be used to encode a region of 3D space.

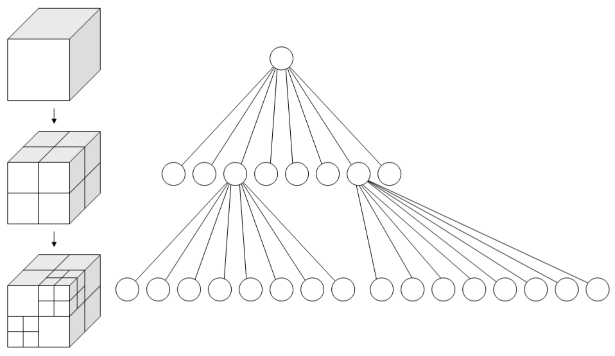


Figure 2.21: Demonstration on how a 3D region of space (left) can be encoded using a (oc)tree (right) [37].

2.3. Nebula Reconstruction

In comparison to the field of nebula visualization, more research has been done on nebula reconstruction. With nebula reconstruction, the goal is to find a plausible particle distribution of a nebula using only telescope images. As was mentioned before, nebulae are too far away to visit them, so we only see them from one point of view. Results in nebula reconstruction make it possible to get more insight into the shape and structure of nebulae so that illustration material for, for example, planetariums and astronomy courses can be created [24].

What all reconstruction methods have in common is that they attempt to find a plausible particle distribution that, when visualized, gives a reconstructed image that is as close as possible to a reference image. More formally, there is a reference image b, a particle distribution (often called a solution) v and a projection system M which takes a solution and converts it into a visualization. Thus we are looking for v where $Mv = b$. Unfortunately the problem is ill-posed; many v's can give the same b when M is applied. Thus constraints have to be applied to make the problem less ill-posed. What constraints are applied differs per work, but most methods assume that the nebula attempted to reconstruct is axial-symmetrical.

Instead of attempting to find the correct v at once, many implementations iterate over many solutions until a v is found that gives a reconstructed image b_{rec} which is sufficiently similar to b. A common approach is to use the difference between b and b_{rec} as an error value to grade the solution v belonging to b_{rec}. This error value can then be used to move in the solution space towards an optimal v.

In Section 2.3.1, we will discuss reconstruction methods used to reconstruct asymmetrical nebula, which are mostly reflection nebulae. Section 2.3.2 will discuss the reconstruction methods that are used for symmetric nebulae.

2.3.1. Asymmetrical Nebula Reconstruction

The only work on reconstructing asymmetrical nebula is done by Linţu et al. [20]. They attempt to reconstruct the 3D dust distribution of reflection nebulae. The appearance of reflection nebulae is dependent on light emission from one or more stars and the scattering and absorption of that light by interstellar dust. These three effects are all taken into consideration during the reconstruction process.

The reconstruction method assumes that there is one central star which emits light. This light is scattered and absorbed by material around the center star. The lighting model takes into consideration light that comes from the center star and that is scattered once and then reaches the viewers eye. In addition to that, the influence of light bouncing (multiple times) between interstellar dust particles before being perceived by the viewer is also taken into account. Again, the problem of finding a v is very ill-posed; there are a lot of possible solutions. Due to scattering, each voxel has a non local effect on the radiance that is received, which does make the problem less ill-posed.

To find a suitable v, this method (along with many other methods including ours) makes use of Powell's method [42]. Powell's method iteratively goes over all entries in v and tries to find the best value for the entry. The difference between b and b_{rec} is computed using the sum of squared differences and that value is used as an error value to grade how correct the found solution is. The error value is used to numerically compute the slope of the error function. This in turn guides Powell's method to a solution that is sufficiently optimal.

However, the order in which the entries in v are optimized by Powell's method heavily influences the quality of the reconstruction and how fast a sufficient solution is found. A naive approach would be to start finding the optimum dust density for an arbitrary voxel. Instead, the reconstruction system of Linţu et al. [20] starts with the voxels that are closest to the center star and working outwards. The scattering happening in these voxels has the largest influence on the accumulated radiance per pixel, as most of the star's light will be absorbed before it reaches the voxels further away. Figure 2.22 shows the result of the reconstruction process.

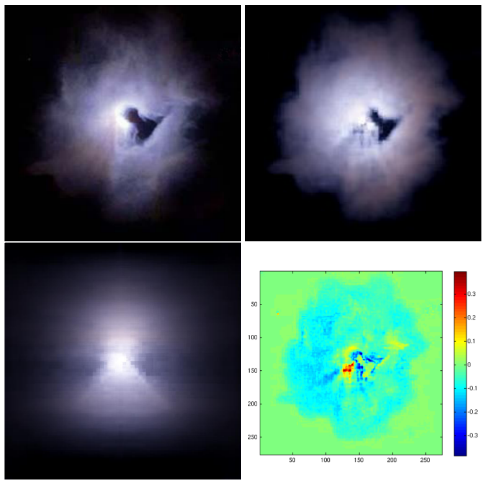


Figure 2.22: In order of appearance: Original image of the nebula, Reconstruction image from the same perspective, Reconstruction image from side perspective, Error estimation of reconstruction vs. original [20].

2.3.2. (Axial) Symmetrical Nebula Reconstruction

Many papers that reconstruct the shape of a nebula exploit their axial- or spherical symmetry to make the process easier. In the volume of an axial-symmetrical nebula, at any point along the axis all information at a certain distance away from the axis is the same. This redundancy can be avoided if we use another method of representation for the volume of the nebula. Instead of storing the data in cuboidal voxels, the information is stored in a ***2D map***. To convert the 2D map to a 3D volume, the 2D map must be rotated around the symmetry axis, resulting in cylindrical voxels. In Figure 2.23 the red grid represents the 2D map. Notice how it is aligned with the symmetry axis of the nebula. The red arrow implies that the 2D map is rotated around the yellow symmetry axis to get the volume of the nebula.

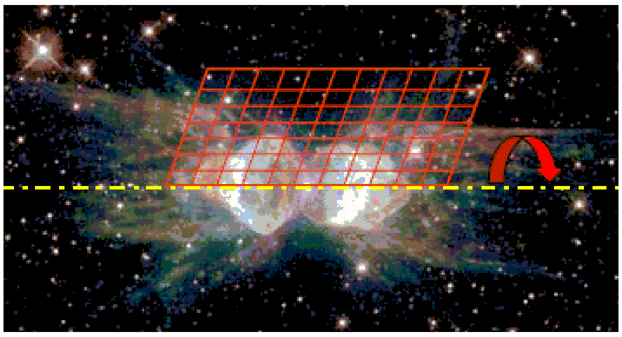


Figure 2.23: A 2D map being rotated around the symmetry axis. Figure from Magnor et al. [23].

This approach was first taken by Magnor et al. [23]. Our method also uses 2D maps to represent the particle distribution of (axial-symmetrical) nebulae. Magnor et al. [23] do not describe in detail how they visualize a 2D map, but presumably they convert the 2D map into a 3D volume with voxels and then visualize that. This in contrast to our rendering method where no conversion step is performed.

Magnor et al. [23] assume that the axial-symmetrical nebula that will be reconstructed is entirely emissive. The influence of scattering and absorption is ignored. Because of this, each color channel of the reference image can be reconstructed independently. In addition to finding the right emissive values for the entries in the (in this case 2D map) v, they also attempt to find the angle of inclination of the nebula.

Similar to Linţu et al. [20], Magnor et al. [23] use Powell's method to find the most suited v. At each iteration step, v is converted from a 2D map into a 3D volume, taking the found inclination angle also into consideration. Then it is visualized using conventional volume-rendering. Next, the resulting b_{rec} is compared to b and the difference between the two is computed using the sum of squared differences. Like before, this error value is used by Powell's method to guide the optimization process. The values in the 2D map may not be negative values. If any entry of the 2D map is negative, a large error value is given to Powell's method. This constraints Powell's method to only look for 2D maps that are physically plausible.

To accelerate convergence, a multi-resolution approach is used. First, a low resolution 2D map is reconstructed (using a low resolution version of the reference image). Once the method converges a higher resolution 2D map is computed (with a corresponding higher resolution version of the reference image). This process happens four times. Results of the reconstruction process can be seen in Figure 2.24.

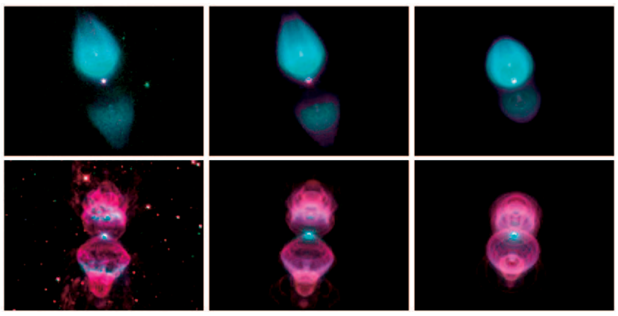


Figure 2.24: Reconstruction of a planetary nebula by Magnor et al. [23]. Left to right: original image, reconstruction viewed from original perspective, reconstruction viewed from "unknown" perspective

The main limitation of this method by Magnor et al. [23] is that it is computationally expensive. The rendering technique that they use does not seem to have special optimizations to aid with rendering axial-symmetrical volumes. They also do not take into consideration possible emptiness around the nebula. Their reconstruction technique takes around 1 day to reconstruct a 128 × 32 2D map.

Both the previously discussed method by Magnor et al. [23] and the next method by Linţu et al. [21] achieve their success by using an optimizer which generates a solution that is rendered and compared to the reference image of the nebula. They call this technique "analysis-by-synthesis" and a diagram of it is shown in Figure 2.25.

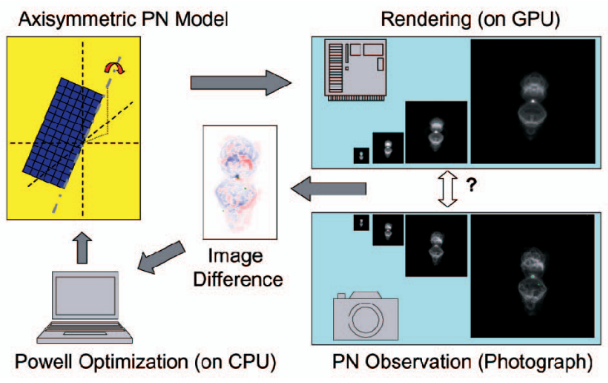


Figure 2.25: Summary of the optimization-reconstruction technique Magnor et al. [23] and Linţu et al. [21]

Linţu et al. [21] continue on the work by Magnor et al. [23]. The principle remains the same: formulate the reconstruction problem as an optimization problem to look for the optimal solution that make the reconstruction image as similar as possible to the reference image of a nebula. However, instead of only taking into account emission, Linţu et al. compute two separate distributions: the distribution of interstellar dust, in which light scatters and the distribution of ionized gasses, which emit light.

The first step of the algorithm is to reconstruct the dust distribution which scatters and absorbs light. For this step an infrared image of the nebula is needed, as this at this wavelength the light cannot penetrate interstellar dust. Thus any (infrared) light that is perceived can only be emitted by the interstellar dust itself, due to scattering, and cannot come from the core of the nebula. This phenomena is shown in Figure 2.26.

Because of this, the dust particles can be interpreted as being emissive. Reconstructing the distribution of emissive dust particles was already done by Magnor et al. [23]. Thus Linţu et al. use the exact same approach (optimization-reconstruction). The only difference is that the final dust distribution should not be

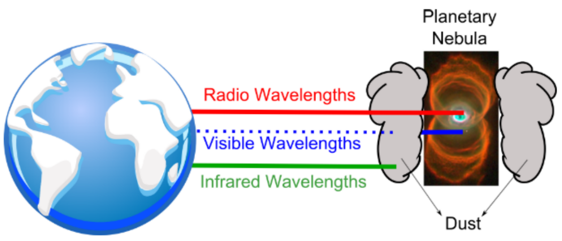


Figure 2.26: Linţu et al. [21] on light absorption by interstellar dust

interpreted as being the dust distribution of emissive particles, but of dust particles that scatter and absorb light.

With this distribution, it is possible to compute the distribution of emissive dust particles. This can also be done using the optimization-reconstruction technique, but the renderer from Magnor et al. [23] is not suited to take into account scattering and absorption of light coming from the center star and from ionized interstellar dust. For this reason, Linţu et al. use a modified version of the renderer by Magnor et al. [22] which is suitable for rendering reflection nebulae. They still assume that the nebula is axis-symmetrical. Unfortunately Linţu et al. do not show visualizations of their reconstructions so we do not know what their reconstructions look like and cannot view the reconstructed nebulae from novel viewpoints.

Wenger et al. [47] use the symmetry in axial-symmetrical nebulae (and point-symmetrical nebulae) by recognizing that, when these nebulae would be viewed from a novel viewpoint, these nebulae would probably appear very similar to how we see the nebulae. These (virtual) novel viewpoints allow us to formulate the problem as a tomographic reconstruction problem. Figure 2.27 shows the placement of these virtual viewpoints. Since the nebula is spherical it will look the same from every angle. Because of this we can place virtual cameras everywhere around the nebula. When a nebula is axial-symmetrical, the virtual viewpoints can only be placed around the symmetry axis of the object.

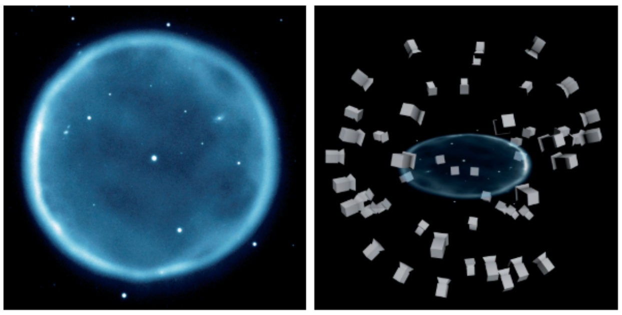


Figure 2.27: Virtual cameras used by Wenger et al. [47]. Left: original image of the nebula, Right: cameras placed all around (an image of) the nebula illustrating the multiple viewpoints

Instead of having one viewpoint b, Wenger et al. [47] define multiple viewpoints b_k. There are also multiple M to convert a solution v to b_k. Wenger et al. [47] do not use a 2D map to encode a 3D volume, but define v as a 3D particle distribution. Given the multiple viewing points, the goal is now to find a v such that $(M_0,...M_{n-1})^T v = (b_0,...,b_{n-1})^T$ holds.

The reconstruction technique by Wenger et al. [47] works well on spherical nebula with a sufficient amount of detail remaining present in the geometry. In comparison, applying the technique to axial-symmetrical nebulae gives slightly worse results, with some details being lost. A significant downside to this reconstruction method is the large amount of computing power that it requires. A cluster of GPUs was used to come to the results from Figure 2.28 and took several hours to compute. The resulting volume consisted of 512^3 voxels. A benefit of reconstructing a 3D particle distribution instead of a 2D map is that the particle distribution can have deformations that break the perfect symmetry. A 3D volume encoded with a 2D map will always be perfectly symmetrical. A downside however, is that it is significantly more difficult to make modifications to the reconstructed particle distribution compared to editing a 2D map.

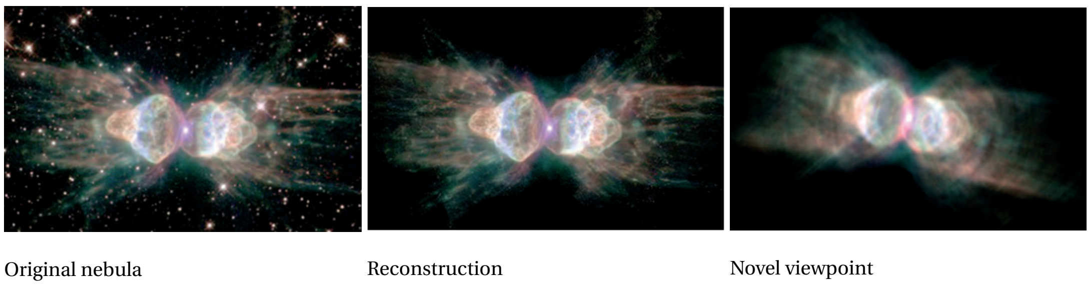


Figure 2.28: Reconstruction of the ant nebula by Wenger et al. [47].

3 Our method

In this Chapter, we describe our improvements on nebula visualization and nebula reconstruction. Everything concerning the visualization of a 2D map will be explained in Section 3.1, where we will discuss how to render axial-symmetrical volumes (Section 3.1.1) and the optimizations that we have developed to accelerate the visualization of nebulae represented by 2D maps (Section 3.1.2-3.1.4). In Section 3.2, we present how we can deform the appearance of the axial-symmetrical nebulae to break their perfect symmetry and make them appear more realistic. Finally, in Section 3.3, we explain our reconstruction approach for reconstructing the 2D map of emissive axial-symmetrical nebulae.

3.1. Rendering

Previous methods already made use of the axial-symmetrical nature of many nebula by defining the volume as a 2D map (Section 2.3). These 2D maps were used for reconstruction, but for rendering purposes the 2D map was always transformed back to a voxel volume. We aim to represent the axial-symmetrical volume with a 2D map which saves storage space compared to using a voxel representation. Another important benefit of using this representation is that it is much easier to make modifications to compared to modifying a 3D volume. Our goal is to optimize the process of visualizing 2D maps such that they can be visualized in real-time.

3.1.1. Axial-Symmetrical Volume Rendering

The techniques that we discussed in Section 2.2.1 are suitable to render an emissive axial-symmetrical nebula. However, as discussed in Section 2.3, the voxel representation of such a nebula contains lots of redundant information, and can be simplified into a 2D map. This 2D map represents the volume by taking a slice through the symmetry axis. Rotating the map around the symmetry axis brings back the original 3D volume. Before, each voxel had a certain emission value, extinction value and color value. Now, each pixel in the 2D density map stores these same properties.

Because we have changed the representation of our volume, the shape of the bounding volume has also changed. Before the shape of the bounding volume was a cube, because the volume was made of cuboidal voxels. When you rotate a rectangle (a 2D map) around a line (the symmetry axis) you get a bounding cylinder. We use Equation 3.7, to compute the intersection points P_{entry} and P_{exit} with the volume. The equation for an infinitely long cylinder aligned along the x-axis with radius r is $y^2 + z^2 = r^2$. We assume that the 2D map, of which we are computing the bounding cylinder, has a width and height of size n. The resulting bounding cylinder will be of length n and its radius will be n.

Equation for a cylinder with radius n

$$y^2 + z^2 = n^2 \tag{3.1}$$

Substituting terms from Equation 2.1

$$(y_O + t \cdot y_D)^2 + (z_O + t \cdot z_D)^2 = n^2 \tag{3.2}$$

Transform into quadratic form

$$at^2 + bt + c = 0 \quad \text{where} \tag{3.3}$$

$$\rightarrow a = y_D^2 + z_D^2 \tag{3.4}$$

$$\rightarrow b = 2y_O y_D + 2z_O z_D \tag{3.5}$$

$$\rightarrow c = y_O^2 + z_O^2 - n^2 \tag{3.6}$$

Computing t_{entry} and t_{exit}

$$t = \begin{cases} \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a} & \text{if } x \geq 0 x \leq n \\ undefined & \text{otherwise} \end{cases} \tag{3.7}$$

When Equation 3.7 has no solutions then there is no intersection with the bounding cylinder (and thus also not with the nebula). Else Equation 3.7 gives two t values: t_{entry} and t_{exit} which correspond to P_{entry} and P_{exit}. When the ray is tangent to the volume Equation 3.7 has only one solution, and we can define $P_{entry} = P_{exit}$.

Like before, we step through the volume from P_{entry} to P_{exit} with a constant stepsize. Sampling the 2D map however, is not as straightforward as sampling the 3D volume, where the 3D position of the sample along the ray could be matched directly to a voxel at that position. Our sample points still have a position of 3 coordinates, but our 2D map is indexed with only two components. Thus we need a conversion function which transforms the coordinates of a sample point S_{xyz} from 3D space to coordinates S_{xy} in 2D map space. As we assume our symmetry axis to be along the x-axis, the x component stays the same, since it represents the x position of the sample point along the symmetry axis. The y component of S_{xy} ($y_{S_{xy}}$), is then defined as the distance of S_{xyz} to the symmetry axis. This definition makes all points that have an equal distance from the symmetry axis and lie on the same orthogonal plane w.r.t the symmetry axis identical, as per the axis-symmetry requirement. We define a conversion function in Equation 3.9.

$$f_{3D \rightarrow 2D}(S_{xyz}) = \tag{3.8}$$

$$f_{3D \rightarrow 2D}(\begin{pmatrix} x_{S_{xyz}} \\ y_{S_{xyz}} \\ z_{S_{xyz}} \end{pmatrix}) = \begin{pmatrix} x_{S_{xyz}} \\ \sqrt{y_{S_{xyz}}^2 + z_{S_{xyz}}^2} \end{pmatrix} \tag{3.9}$$

With Equation 3.9 we know how to look up values from our 2D map at each sample step. We present a pseudo-code implementation on how to do axial-symmetrical Volume Rendering in Algorithm 2.

Algorithm 2 Pseudo-code implementation of axial-symmetrical Volume Rendering

```
Require: $O, D \leftarrow$ Origin and direction of ray
Require: $t_{entry}, t_{exit} \leftarrow$ Entry point and exit point of ray with bounding cylinder
Require: $M \leftarrow$ 2D map
  $C_{acc} \leftarrow$ (accumulated radiance) 0
  $A_{acc} \leftarrow$ (accumulated extinction) 0
  for $t \leftarrow t_{entry}, t_{exit}$ do
      $S_{xyz} \leftarrow O + t \cdot D$
      $S_{xy} \leftarrow f_{3D \rightarrow 2D}(S_{xyz})$
      $c, a, b \leftarrow M[S_{xy}]$                ▷ Retrieve color, emission and extinction values
      $C_{acc} \leftarrow C_{acc} + (1 - A_{acc}) \cdot (a * c)$
      $A_{acc} \leftarrow A_{acc} + b$
  end for
```

While using the 2D density map for rendering greatly reduces the amount of memory necessary to store the volume, a lot of sampling steps per ray are still necessary to find the accumulated radiance for each pixel. To improve the rendering performance, we will describe our optimizations to reduce the amount of sampling steps taken when using our axial-symmetrical volume rendering method.

3.1.2. Sparsity in 2D maps

Figures 3.1, 3.2 and 3.3 show what the 2D map of axial-symmetrical nebulae can look like. Notice that these 2D maps contain empty regions (mainly around the main particle distribution) and these empty regions will correspond to empty volume if the 2D map is converted into a 3D volume. Performing a sampling step and a blending step is unnecessary when we are in a region that is empty, since empty space will not contribute radiance to pixels in the output image. If we can check whether a region is empty, we can prevent an unnecessary memory fetch (which leads to stalls in the instruction pipeline, i.e., we have to wait before the memory is loaded). In addition to that, we simply do not need to perform the computation that is needed to blend the observed radiance thus far with the sampled radiance.

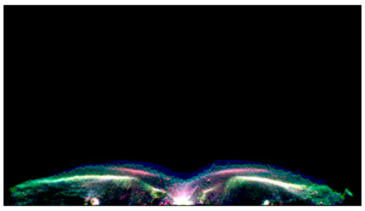


Figure 3.1: 2D map of M2-9 nebula

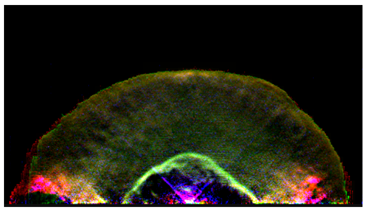


Figure 3.2: 2D map of NGC 6826 nebula

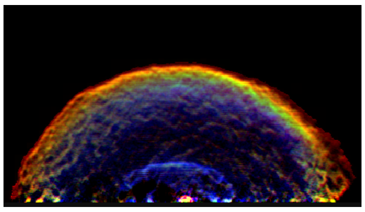


Figure 3.3: 2D map of IC 418 nebula

We need a suitable data structure to know which regions of space are empty and which ones are not. As we have discussed in Section 2.2.2, in Volume Rendering it is common to use octrees to exploit uniform regions present in a volume. In our case, our model is represented by a 2D map and not a 3D grid. As can be seen in the Figures 3.1, 3.2 and 3.3, the only uniform regions that are present in a 2D map are the empty regions bordering the outside of the nebula. The distribution of particles is in most nebulae centered around the symmetry point or symmetry axis. There are barely any empty regions in the nebula itself. A data-structure that fits this distribution and can easily encode empty portions on the outside, is a height-map, which we will detail in the following.

Given a 2D map M and a sampling point S_{xy}, we want to know if S_{xy} is in a region of M that is empty. To describe this formally, we use Equation 3.10.

$$\forall x \in M, M_{x,y} = \begin{cases} \text{empty} & y \geq y_{\emptyset} \\ \text{non-empty} & y < y_{\emptyset} \end{cases} \tag{3.10}$$

What Equation 3.10 says is that for the xth column in M, there is a y which we call $y_{\emptyset}$ whereby all pixels above that $y_{\emptyset}$ are guaranteed to be empty and all pixels below $y_{\emptyset}$ are (probably) non-empty. We say probably, because it might be that only one pixel is non-empty and all pixels below that pixel are actually empty. Given the distribution of particles in axial-symmetrical nebulae that we just discussed, this is unlikely, but it can happen. The set of all $y_{\emptyset}$ values of M we will call its heightmap H. Note that we only have to go over the columns of M once to find what parts of the volume corresponding with M are empty. We can use an array to store and access H. Since we only have n columns in M, we can store H in an array of size n.

To find out whether S_{xy} is in a region that is empty, we only need to compute which column of M we are currently in (which we can do using the x component of S_{xy}) and compare the y component of S_{xy} with the $y_{\emptyset}$ corresponding to the current column. If the y component of S_{xy} is above $y_{\emptyset}$ then we know that S_{xy} is at a point in M that is empty.

We call the optimization that we have introduced the **emptiness-check**. Doing an emptiness-check prevents unnecessary work like querying M and doing radiance accumulation. We make use of the emptiness that can be present in the 2D map of axial-symmetrical nebulae.

3.1.3. Global maximum in 2D maps

In the previous section, we described how we can prevent the sampling of regions which are empty. Sampling these regions is unnecessary and wastes computation time. Still, for every step along a ray that we take in our rendering algorithm we have to check whether the current position is empty. Even though we are preventing unnecessary memory fetches and computations, this checking requires us to do a lookup in H and a check whether our current position is below the corresponding $y_{\emptyset}$. Doing so is faster than not checking whether a region is empty, but it is not free. In the extreme case, if we were to visualize a 2D map that is completely empty then we would still do these computations and this feels wasteful, as the radiance that is accumulated along a ray that only goes through empty space is obviously zero.

Now, we want to know whether a ray that goes through the volume never intersects with any non-empty volume. This is true for rays where the distance between the ray and the symmetry axis never becomes less than any of the $y_\emptyset$ in H. The point at which the ray is closest to the symmetry axis we call t_{min}. We can find t_{min} by using our knowledge of how rays traverse 3D space and the relation that this has to our position in the 2D map. In Section 3.1, we defined a conversion function $f_{3D\rightarrow 2D}$ (Equation 3.9) to take a sampling point S_{xyz} in 3D space to a sampling point S_{xy} in 2D space. To make the relation between the trajectory of a ray in 3D volume space and 2D map space more clear, we present Figure 3.7 where we plotted the trajectory of multiple rays going through the bounding cylinder in 3D space and showing the path that they follow on a 2D map.

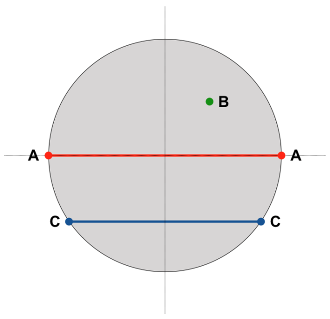


Figure 3.4: Side view of rays going through a cylinder

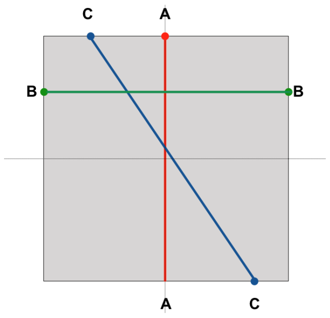


Figure 3.5: Top view of rays going through a cylinder

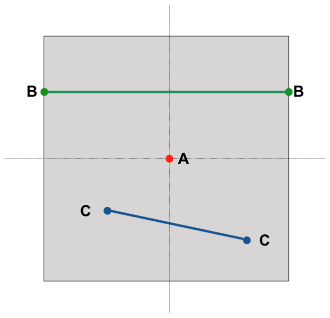


Figure 3.6: Front view of rays going through a cylinder

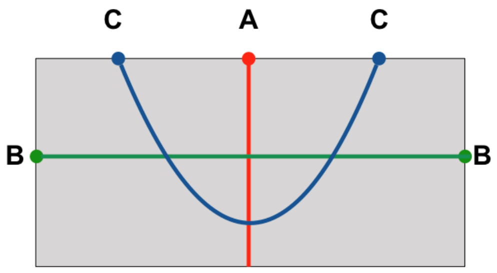


Figure 3.7: Trajectory of rays in figures above plotted on a 2D map.

The red ray travels straight through the bounding cylinder and intersects with the symmetry axis. The green ray travels parallel to the symmetry axis. Finally, the blue ray is angled w.r.t. the symmetry axis, but does not intersect with the symmetry axis.

Each of these rays have a point at which they are closest to the symmetry axis: t_{min}. Obviously this point cannot lie before the entry point or after the exit point. Therefor t_{min} is clamped between t_{entry} and t_{exit}. It might be that the ray will be even closer to the symmetry axis before t_{entry} or after t_{exit}, but this point will be outside of our bounding cylinder. Thus the following holds for t_{min}:

$$\forall t, t_{min} = \begin{cases} t_{entry} & t < t_{entry} \\ t_{exit} & t > t_{exit} \\ \text{Some value for } t & t_{entry} < t < t_{exit} \end{cases} \tag{3.11}$$

The corresponding distance of the ray and the symmetry axis at t_{min} is denoted as $y_{t_{min}}$. In Figure 3.7, we see in 2D map space that the red ray intersects with the symmetry axis and thus $y_{t_{min}}$ for the red ray is zero. The green ray remains parallel to the symmetry axis, so its distance to the symmetry axis is constant. We could arbitrary pick any point along the ray and its distance to the symmetry axis would be $y_{t_{min}}$. Finally, for the blue ray we see that it does not intersect with the symmetry axis.

Notice that the shape of the blue ray in Figure 3.7, is somewhat shaped like a parabola. This is caused by how $y_{S_{xy}}$ is defined in Equation 3.9. In Equation 3.12, we will derive the term $\sqrt{y^2_{S_{xyz}} + z^2_{S_{xyz}}}$ to a quadratic polynomial that is more recognizable. To do so, we will substitute terms from Equation 3.9 with terms from Equation 2.1.

$$\begin{aligned} \sqrt{y_{S_{xyz}}^2 + z_{S_{xyz}}^2} &= Y \\ y_{S_{xyz}}^2 + z_{S_{xyz}}^2 &= Y^2 \\ (y_O + t \cdot y_D)^2 + (z_O + t \cdot z_D)^2 &= Y^2 \\ (A + t \cdot B)^2 + (C + t \cdot D)^2 &= Y^2 \\ A^2 + 2AB \cdot t + B^2 t^2 + C^2 + 2CD \cdot t + D^2 t^2 &= Y^2 \\ (B^2 + D^2) \cdot t^2 + (2AB + 2CD) \cdot t + (A^2 + C^2) &= Y^2 \\ \alpha t^2 + \beta t + \phi &= Y^2 \end{aligned} \tag{3.12}$$

The final term $\alpha t^2 + \beta t + \phi$ is a quadratic polynomial and computes the squared distance between the ray and the symmetry axis (Y^2) at step t. From this formula we can derive at which t the distance between the ray and the symmetry axis is minimum. Minimizing Y^2 is equal to minimizing the absolute value of Y, which is exactly the (minimal) distance we need. We take the derivative of the polynomial in Equation 3.12 to find t_{min}. This is shown in Equation 3.13.

$$\begin{aligned} f(t) &= \alpha t^2 + \beta t + \phi \\ df(t) = t_{min} &= \frac{-\beta}{2\alpha} \end{aligned} \tag{3.13}$$

Given that the distance that a ray will be from the symmetry axis is expressed with a parabola, we know that any points that lie before or after t_{min} can only be the same distance from the symmetry axis or further away. It can occur that α is zero due to the direction vector D. This corresponds a ray parallel to the symmetry axis, like the green ray in Figure 3.7. As we said before, t_{min} is clamped between t_{entry} and t_{exit} and in the case of $\alpha = 0$ it can be any valid t.

Any ray with a $y_{t_{min}}$ that is larger than the largest $y_{\varnothing}$ value in H, will never intersect with any volume in M. To determine the largest $y_{\varnothing}$ value in H, we need to go over all $y_{\varnothing}$ values and store the largest one. We call the largest $y_{\varnothing}$ value H_{max}. We can discard the rays for which $y_{t_{min}} > H_{max}$, because sampling them would not accumulate any radiance. This test whether we can discard a ray entirely is done before the actual sampling starts. We will refer to this optimization as the **global-maximum-check**.

3.1.4. Using a variable step size

At the moment, we traverse the nebula at a constant step size; the 3D distance between a sample point and the next sample point always remains the same. When the *global-maximum-check* of the previous section returns true, it is as if we traverse through the nebula with one large step from t_{entry} to t_{exit}. Being able to perform such a large step will probably be uncommon, because the conditions under which we can do such a step are strict. It is entirely possible that a ray, while going below H_{max}, still has no intersection with any volume in M. It is also possible that only a part of the ray intersects with non-empty volume. Ideally, we would want to go through the empty parts of the volume with as large of a step as possible. The size of this step, ω will determine whether we can take this step or not. In this section we shall discuss how we can know whether it is safe to perform a step of size ω.

We can do such an adapted step of size ω when we know for sure that there is only empty space between the point from which we start (t) and the point at which we want to land $(t + \omega = t_\omega)$. Before, we used H_{max} but this is the global maximum of H. The ray can fall within a section of M that does not have any volume which is of height H_{max}. Ideally, we want to know the maximum of H for the section of M that the ray falls into. This subsection of M also has a largest $y_{\varnothing}$, which is a local maximum. H stores enough information to know what the local maximum is of a section, but it is not structured to retrieve this information easily. We will therefor construct a binary tree over the height map H, such that we can look up this information in constant time. Each node in the tree stores the local maximum of a section of H. This means the first layer stores the global maximum, the second layer stores the local maximum of the left and right side of M, the third layer stores the local maximums of M if it is sectioned into quarters, etc. This tree can be built from H

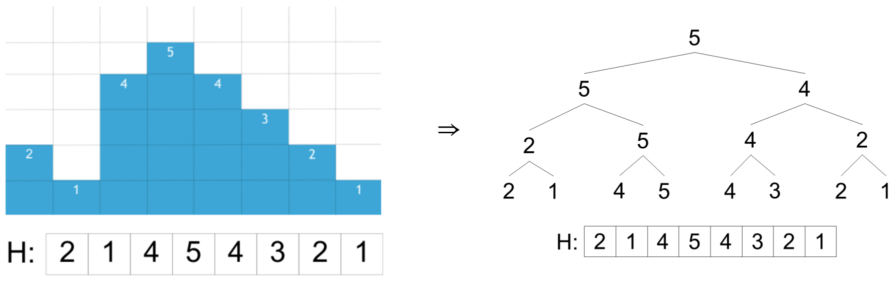


Figure 3.8: The 2D map (left) is transformed into a height map (right). The height map has a tree-like shape.

by placing all $y_\emptyset$ values in the leaves and building it bottom-up. We show the transformation from H as an array to H as a tree-like structure in Figure 3.8.

Now, if $\omega = t_{exit} - t_{entry}$ then we know that the lowest part of the ray is between t and t_ω is t_{min}. If we define ω to be of arbitrary size then the lowest point on the ray between t and t_ω (t_{lmin}) is described in Equation 3.14.

$$t_{lmin} = \begin{cases} t_\omega & t < t_\omega < t_{min} \\ t_{min} & t < t_{min} < t_\omega \\ t & t_{min} < t < t_\omega \end{cases} \tag{3.14}$$

This relationship holds due to the quadratic Equation 3.13 and the parabola that it defines. t_{min} is the lowest point on this parabola. For a subsection of a ray, the point t that will be closest to the symmetry axis will always be the point closest to t_{min}.

We now know the point at which the ray is closest to the symmetry axis if we want to perform a step of size ω. To know whether it is safe to do this step, we need to compute in what section of H the ray falls. Using the distance between the x component of P_t and P_{t_ω}, we can compute the smallest section that both points fall into. If the height-tree value of that section is smaller than t_{lmin}, it is safe to perform the step. We do not want to use a section that is larger than necessary, because a larger section corresponds to going one level up in the tree H and then there might be a possibility that the found maximum is a maximum of a section that the ray does not fall into.

Now, the value that we select for ω will influence how often we can perform a step of size ω. If ω is large then performing a step of size ω can only happen when large parts of the ray go through empty space. Picking a small value for ω means that we only skip parts of the ray and there is more checking that we have to do to see if we can perform the step. We propose a hierarchical approach where we test a value for ω and if that is not successful, we try a smaller value for ω. If we conclude that we cannot perform a step with the smallest ω then we sample all points between t and t_ω as normal. At each sampling step the *emptiness-check* can still be performed. The optimization that we have just discussed is the last optimization that we introduce and we refer to it as the **ω-step-check**. In Chapter 5, we experiment with different values for ω to find which value(s) are best suited when visualizing different axial-symmetrical nebulae.

3.2. Reintroducing asymmetry

So far, we have only focused on creating a completely symmetric rendition of a nebula. This is an inherit trait of our rendering pipeline. Our visualization method rotates a 2D image (a 2D map) around an axis. The resulting volume is perfectly symmetrical around this axis. Yet, if we look at nebulae in space, we see that they are not perfectly symmetrical. Nebulae often have small shape and color deformations, which makes their appearance less uniform. We have shown a couple of examples of nebulae, which have this characteristic. Figure 2.10 of the NGC 6826 nebula is a great example of this.

Renditions of nebulae that are entirely symmetrical can look artificial. To make our visualizations appear more realistic we made changes to our rendering pipeline such that the nebulae that we generate look more asymmetrical. We did this while still representing the volume of the nebulae with a 2D map. We allow the appearance of a nebula to be modified in two ways: using color deformations and using shape deformations. It is our goal to ensure that the optimizations that we discussed in Section 3.1 remain applicable.

3.2.1. Color deformations

As can be seen in Figure 2.10, nebulae are not uniform in color. Some parts emit more light than other parts. To mimic this behaviour, we are going apply color deformations using smooth noise functions. A smooth noise function takes as its arguments a position in (in our case) 3D space. At each sampling step, we use the position in the scene to lookup a noise value. This noise value is simply a floating point value, and we define it to be between 0 and 1. The important property that smooth noise has, which makes it useful to apply realistic color deformations, is that noise values that are close to each other in space are also similar in value.

The color deformation that we apply reduces the amount of emission and extinction at a certain point within the volume. The reduction in emission corresponds to less light being emitted from that point. In real nebulae this corresponds to material being less hot than material around it. A reduction in the extinction factor of a point within the nebula means that it will not block as much light coming from points behind it. Absorption is caused by dense interstellar dust (Section 2.1.2) and a decrease in absorption corresponds to a decrease in the density of interstellar material. When dust particles are less packed together then the probability that they will block light is lower.

There are different types of noise functions that we can use. Picking the right sort of noise is mostly an artistic choice. A common noise function is Simplex noise by Ken Perlin [41]. It is used often in the context of modeling some natural occurring phenomena, such as generating terrain in video games or modeling clouds [19]. Other types of noise functions are Turbulent noise, Fractal noise and Cellular noise. Figure 3.9 shows different types of noise and what the result is when applying them to a volume.

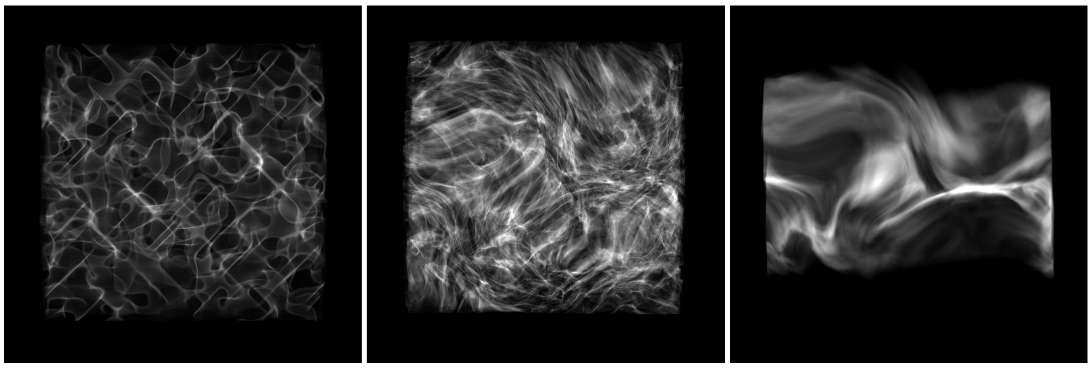


Figure 3.9: Different types of color deformations applied to an axial-symmetrical volume

3.2.2. Shape deformations

In order to further improve the appearance of the nebulae, we also consider the fact that real nebulae are not uniform in shape. The shape of nebulae is axial symmetrical on a macro level, however, phenomena, such as supernova explosions and stellar winds, make the shape of a nebula generally less regular. This breaks the axial-symmetry. See Section 2.1.1 for more information on this.

It would be infeasible for our system to take into account the effect of all the natural phenomena that influence the shape of a nebula by way of simulation. What we can do is formulate some rough approximations in order to come close to the shape of the nebulae that we observe in space. In the nebulae that we just discussed, we found that the cause of their deformations was mostly due to their center star that was exuding forces, which distorted the shape of the nebula. Our system has no notion about stars, their position and the amount of force that they have. Instead we define a field of forces that acts around the symmetry axis of our nebula and pretend that the symmetry axis is pushing matter outwards. When no deformation is applied, the force that the symmetry axis exudes is at maximum power. Thus, the more deformation that is applied, the less points will be forced outwards. This might feel a bit counter intuitive, but we can only move sample points towards the symmetry axis. If we were to move sample points outwards, their distance from the symmetry axis would be larger than the value that is stored in H for the respective column of M that the sample points falls into and render our acceleration structure invalid. To ensure that radiance that is near the symmetry axis (such as the center star) is not completely moved "below" the symmetry axis, we use the distance that a sample point is from the symmetry axis as a normalization value. Points that will be far away from the symmetry axis will be moved inwards, but points that are close to the symmetry axis will stay more at the same position.

The amount of force that is applied is stored in a 2D map called the displacement map. We access this displacement map by mapping the x component of a sample point to a column in the displacement map and the y and z components of a sample point as an angle w.r.t. the symmetry axis and use this angle to access a row in the displacement map.

To ensure that the deformations that occur due to the displacement map are smooth, we again make use of smooth noise.

3.3. Reconstruction

In Section 3.1, we discussed how we can optimize the rendering of an axial-symmetrical nebula. We analysed our data and based on the outcome we made some assumptions (like in Section 3.1.2, where we found that 2D maps of nebulae are often sparse). These assumptions formed the basis for our optimizations. As the reconstruction process makes extensive use of rendering the nebula, all of these optimizations also help to accelerate this part of our method. In addition to that, we will try to simplify the reconstruction task even further to increase the performance of the reconstruction.

The core of our reconstruction technique is similar to other nebula reconstruction techniques discussed in Section 2.3. The goal is to find a 2D map representing a 3D axial-symmetrical volume, such that when this volume is rendered from the original telescope view point, the difference between the resulting image and the reference image of the nebula is minimized.

The structure of this Section is as follows. In Section 3.3.1 we briefly discuss what technique our method shares with the other reconstruction techniques from Section 2.3.2. In the next section (Section 3.3.2) we describe our attempt at simplifying the reconstruction task by assuming that the symmetry axis of the nebula is orthogonal to the viewing direction. We will see that the sparsity around a nebula will again play a role in the optimizations that we can apply.

3.3.1. Similarities with other reconstruction methods

First, we want to discuss which parts of our reconstruction method are also found in other reconstruction techniques. We use the the analysis-by-synthesis strategy (Figure 2.25) that was first proposed by Magnor et al. [23] to also reconstruct axial-symmetrical emission nebulae.

Error minimization Magnor et al. [23] and Linţu et al. [20, 21] use the Powell's non-linear optimization method. We also adopt the same algorithm. As we explained in Section 2.3, Powell's method goes over each entry in M and tries to find a value for that entry until the error (the difference between b and b_{rec}) is sufficiently low. When it has found a suitable value for one 2D map entry it will try to optimize the next entry. The more entries, the more configurations must be evaluated before a sufficient solution is found. It is therefor desirable that the number of variables that the solver must find a value for is as low as possible.

Hierarchical approach We employ a hierarchical approach when reconstructing a nebula. We first start with a lower resolution version of the reference image and then progressively increase the resolution until a reconstruction of the desired resolution is achieved. After each reconstruction step, we upscale the found M and use the up-scaled M as an initial solution for the next reconstruction step. This speeds-up the reconstruction process and it prevents being stuck in a local minimum.

Completely emissive volume Assuming that absorption and scattering plays no role in the appearance of the nebula allows us to simplify the reconstruction task significantly. Ignoring scattering means that we have to consider self-emission as the only means by which radiance can be contributed. The radiance that is received by us is then only dependant on all radiance that lies on the camera ray.

Ignoring absorption simplifies the the reconstruction task even further. If we look at Equation 2.3 we see that each color intensity is attenuated by the same absorption coefficient. This means that, for an entry in M, we must reconstruct the color intensity as well as the absorption coefficient at the same time. Given that we want to reconstruct a colored image of a nebula, this means that we must optimize four variables per entry in M.

This is unfortunate, because the more variables that must be optimized, the more complex the optimization task is. Instead, if we assume that there is no absorption coefficient then the three color intensities do not have a common dependency. Thus we can reconstruct the red, green and blue color channels separate from each other.

Perfect symmetry While our rendering method does allow for the (re)introduction of color and shape deformations, we do not attempt to capture these details during reconstruction. All known reconstruction techniques that do consider shape deformations do not work with a system that relies on a 2D map for the volume definition of the nebula. Instead those systems reconstruct a 3D volume description.

We investigated if we could reconstruct the 2D displacement map, but unfortunately the results were not promising. This could be an interesting topic to explore further in the future (Section 7). With the way displacement is currently implemented (the deformation that is applied is the same for each color channel of a pixel in M), it would mean that there would be a dependency between the color intensities of a pixel in M. This would result in the fact that the shape deformation and the colors in M would need to be reconstructed at the same time.

3.3.2. Assuming a fixed axis alignment

The basis for our optimizations is that we assume that the symmetry axis has an inclination of 0°. In other words, when we view the nebula from Earth, the symmetry axis is perpendicular to the direction in which we are viewing it.

Assuming an inclination of 0°means that a column of rays in the image plane corresponds to a column of map entries in M. This means that we can reconstruct a column in M independently from another column in M. If the nebula were to be angled then this would not be the case. This greatly decreases the number of variables that need to be optimized for at once.

Another reduction to the number of variables that we have to solve for, when we assume an inclination of 0°, has to do with the sparsity found around nebulae. Similar to how we avoided sampling regions that are known to be empty, we should try to avoid finding map entries that are known to be empty. During rendering, we used H to know which regions of space were empty. Now, we do not have a heightmap, but we can compute which map entries are going to be empty by using a method that is similar to how we compute a heightmap. We go over each column in the reference image and look at what distance from the symmetry axis the first non-empty pixel is. Anything above this pixel we can expect to be empty. This is similar to how we found all $y_{\emptyset}$ in H. We can eliminate quite a few variables in this way.

In addition to knowing which map entries will be empty, we will also know which pixels in the image of the reconstruction will be empty. Using our theory from Section 3.1.4, we can skip over the empty space immediately and do not have to sample the empty map entries that lie before and behind the nebula. The size of the step that we do to skip over the empty space is given by the amount of empty map entries.

Finally, due to the axial symmetry and that the inclination angle is 0°, the accumulated radiance that lies between t_{entry} and t_{min} is the same as the radiance between t_{min} and t_{exit}. When computing how much radiance is received by a pixel in the image plane, we only need to retrieve and accumulate the radiance between t_{entry} and t_{min}. The radiance received by the pixel in the image plane is twice this value.

4 Implementation

This Chapter will discuss the workings of the applications that we have created to visualize and reconstruct nebula. We will give details concerning the use of programming languages, libraries and API's used. Our nebula visualizer is discussed in Section 4.1. It allows the user to create and manipulate a 2D map and apply deformations using smooth noise. With the program it is possible to visualize a nebula using its 2D map, modify a 2D map to edit or create a new nebula and apply color and shape deformations to nebula using different smooth noise functions. We discuss the user interface of this program in Section 4.1.1 and dive deeper into its architecture in Section 4.1.2. Our nebula reconstruction program is discussed in Section 4.2, whereby we discuss its interface and architecture in Section 4.2.1 and Section 4.2.2 respectively. This program can be used to reconstruct a 2D map from a picture of a nebula. The visualizer and reconstructor are separate programs which could be integrated, but due to time constraints it was easier to keep the two projects separate.

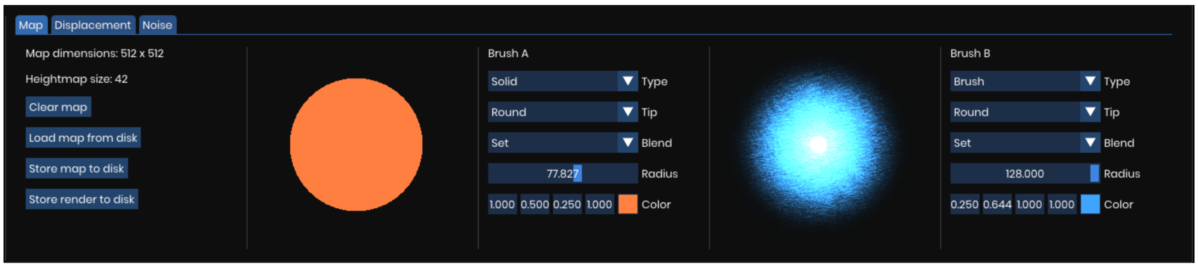
4.1. Nebula Visualization

The goal of our visualizer program is to provide a user interface, which should allow the user to create new nebulae and view and edit existing nebula (reconstructed with our reconstruction program). The user is able to apply shape and color deformations in a simple manner to make the appearance of the nebulae more realistic. The tool is not intended as a physics simulator, but to make it easier to model axial-symmetrical nebulae with a realistic appearance.

4.1.1. Interface

An overview of the interface of our application is shown in Figure 4.1.

The 2D map, displacement map for shape deformations and 3D noise field for color deformations are shown to the user. The user is able to draw on the 2D map with the corresponding volume updating in real-time. The user can manipulate the 2D map with different types of brushes to draw in different styles. They are also able to choose brush properties such as the color of the brush (which will influence the emission and absorption) and things such as the radius of the brush. An overview of the options is shown in Figure 4.2.



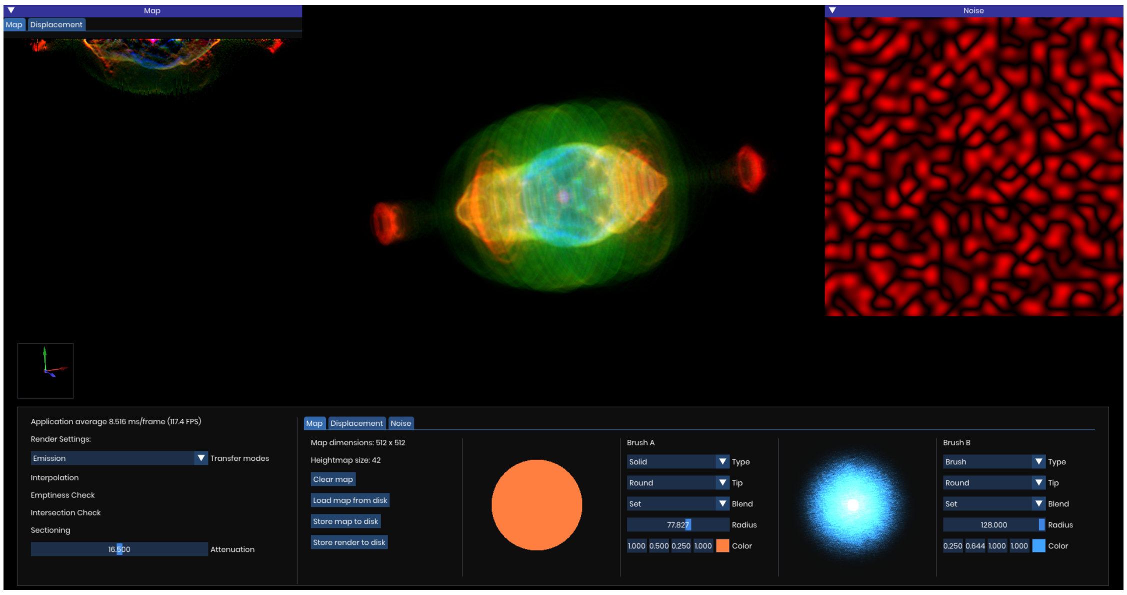
Map dimensions: 512 x 512
Heightmap size: 42

Brush A
Type: Solid
Tip: Round
Blend: Set
Radius: 77.827
Color: 1.000 0.500 0.250 1.000

Brush B
Type: Brush
Tip: Round
Blend: Set
Radius: 128.000
Color: 0.250 0.644 1.000 1.000

Figure 4.2: Overview of the controls that the user has to modify the 2D map

Smooth noise functions are used for color and shape deformations (see Section 3.2). Figure 4.3 shows the different settings exposed to the user to control how the noise is generated. Separate noise settings are used



Application average 8.516 ms/frame (117.4 FPS)
Render Settings:
Emission ▼ Transfer modes
Interpolation
Emptiness Check
Intersection Check
Sectioning
16.500 Attenuation

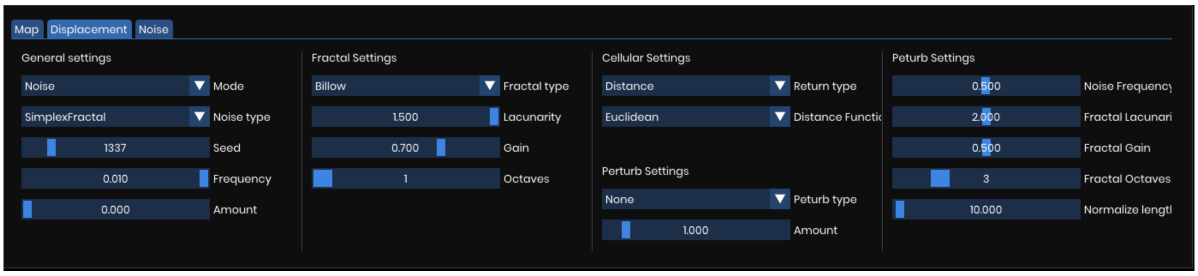
Map Displacement Noise
Map dimensions: 512 x 512
Heightmap size: 42
Clear map
Load map from disk
Store map to disk
Store render to disk

Brush A
Solid ▼ Type
Round ▼ Tip
Set ▼ Blend
77.827 Radius
1.000 0.500 0.250 1.000 🟧 Color

Brush B
Brush ▼ Type
Round ▼ Tip
Set ▼ Blend
128.000 Radius
0.250 0.644 1.000 1.000 🟦 Color

Figure 4.1: Overview of the visualization application

for the color and shape deformations. The user is able to select different types of noise functions and define their parameters. In addition to the different parameters, the user is also able to draw the displacement map. This is similar to how drawing works for the 2D map. These controls are not shown in Figure 4.3 but they are similar to the controls shown in Figure 4.2.


Map Displacement Noise

General settings
Noise ▼ Mode
SimplexFractal ▼ Noise type
1337 Seed
0.010 Frequency
0.000 Amount

Fractal Settings
Billow ▼ Fractal type
1.500 Lacunarity
0.700 Gain
1 Octaves

Cellular Settings
Distance ▼ Return type
Euclidean ▼ Distance Functio

Perturb Settings
None ▼ Peturb type
1.000 Amount

Peturb Settings
0.500 Noise Frequency
2.000 Fractal Lacunari
0.500 Fractal Gain
3 Fractal Octaves
10.000 Normalize lengtl

Figure 4.3: Overview of the settings used to compute the smooth noise for color and shape deformations

Finally, there are buttons so that the user can store an image of the nebula that they have visualized, load and store 2D maps, and load and store noise settings.

4.1.2. Architecture

Our visualization program is written in C++ [13] and uses `OpenGL` [11] for the graphical user interface (GUI). The volume rendering is done using CUDA [38].

Libraries A number of libraries are used to simplify the development process. The library `ImGUI` [5] allows for easy UI development and is also extendable. `ImGUI` interfaces with `OpenGL` to draw the UI elements on screen. In addition to that, the rendering pipeline using `CUDA` also interfaces with `OpenGL` and with `ImGUI` such that the output image can be shown using `ImGUI` to the user.

To compute smooth noise values for shape and color deformation, our application uses the `FastNoiseSIMD` [40] library to compute the noise. The library makes use of the `SIMD` programming technique to achieve better performance. On top of that, `FastNoiseSIMD` supports many kinds of smooth noise, such as `Perlin noise`, `Simplex noise` and `Cellular noise`. It also supports fractal deformations and can compute noise in 2D and 3D. Unfortunately there is a downside to using `FastNoiseSIMD` and that is that it runs entirely on the CPU. We need the noise data on the GPU and it is expensive to transfer the data from the CPU. The overhead is acceptable, as we do not need to transfer the data every frame, but only when the data is modified.

To ensure that the evaluation of the noise functions is efficient, we divide up the workload such that it can be evaluated in parallel on multiple CPU threads. To help with generating multi-threaded code, we use the `OpenMP` [39] library.

GPU Programming Model Using GPUs to accelerate volume-rendering tasks is common practice (as discussed in Section 2.2.2). We used CUDA from NVIDIA [38] to harness the computing power of GPUs. The workload is still as parallelizable as before. The computation of the radiance that is perceived by a pixel in the output image can be computed independently of all other pixels. The modifications that we have made to the volume-rendering algorithm to work for axial-symmetrical volumes described by 2D maps do not prevent us from using a GPU to help with the workload. In addition to that, our optimizations also do not interfere with the GPU programming model.

4.2. Nebula Reconstruction

4.2.1. Interface

Our reconstruction program does not have a user interface. It is accessible through the command line. It takes an image of a nebula as an input. There are no dimension constraints that the user has to take into consideration when supplying an image. The only thing to ensure is that the symmetry axis of the nebula is aligned correctly, meaning that it is completely horizontal and at the image center (equal distance away from the top and bottom of the image). We also assumed that any background stars are removed from the reference image.

4.2.2. Architecture

Like our visualization program, the reconstruction program is written in `C++`. With our reconstruction approach, each column of the 2D map can be reconstructed separately. In addition to that, each color channel can also be reconstructed separately. This makes the workload very parallelizable. Like with the other reconstruction approaches (Section 2.3.1), we use Powell's method to find a solution that, when visualized, gives a result that is as close to the reference as possible. We need to employ $n \times c$ instances of Powell's method to find a solution for each of the n columns of the 2D map whereby each column has c color channels. We first experimented with running the reconstruction process on the CPU or having parts of the reconstruction process run on the CPU and other parts on the GPU. With our hardware configuration, we found that running the workload entirely on the GPU gave the best performance.

First, the reference image is placed in GPU memory such that every kernel invocation has access to it. We also reserve memory for the reconstructed 2D map. Next, we launch $n \times c$ blocks with 1 thread. It is more common for GPU programs to use a higher number of threads per block. This works under the condition that the work that each thread must perform is similar. With testing we came to the conclusion that the work that must be performed to reconstruct a column of the 2D map was so heterogeneous that we did not benefit from having multiple threads in a single block performing this task. This programming model leaves the GPU underutilized, which is something that future research could address.

5 Results

In this chapter, we will share visual and quantitative results of our methods. The visual results of our rendering method from Section 3.1.1 and Section 3.2 are shown in Section 5.1.1. A quantitative analysis of the performance of our rendering method is given in Section 5.1.2. Finally, the results of our reconstruction method from Section 3.3 is given in Section 5.2.

5.1. Rendering

We have evaluated our rendering method in two ways: visually and quantitatively. To do so, we have made a selection of nebulae and visualized these with our rendering method. We used 2D maps from nebulae reconstructed using our reconstruction method and then modified the nebulae in our application to add deformations to make them slightly asymmetrical. We applied shape and color deformations in a way that we found both aesthetically pleasing and physically plausible. The nebulae were selected because they have an interesting shape that could influence the performance of our rendering method. The effect of their shape on the performance of our rendering method will be discussed in Chapter 6.

5.1.1. Visual results

We will now present several nebulae visualized using our rendering method and modified using shape and color deformations.

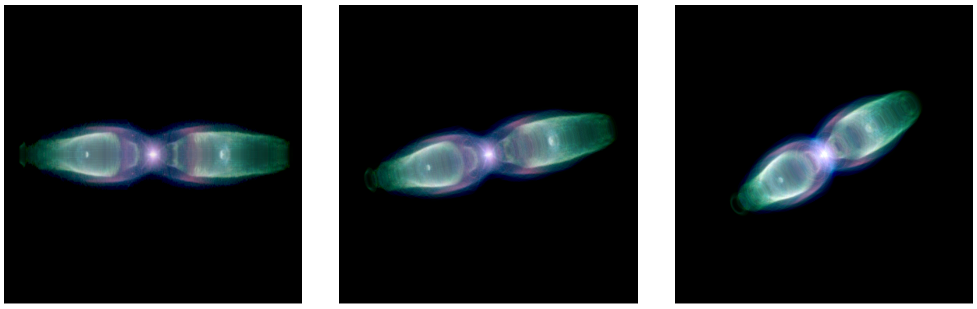


Figure 5.1: M2-9 nebula without deformations

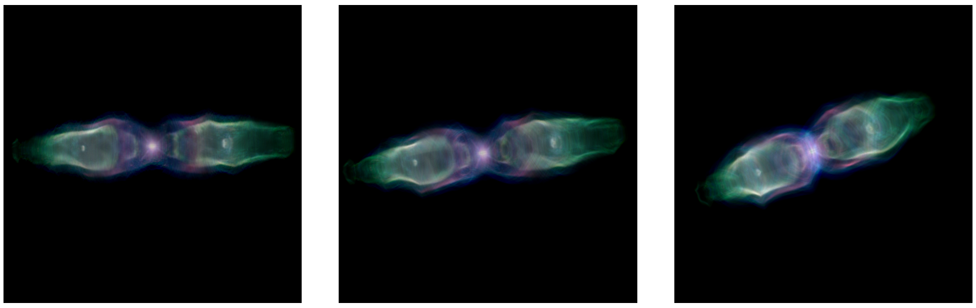


Figure 5.2: M2-9 nebula with deformations

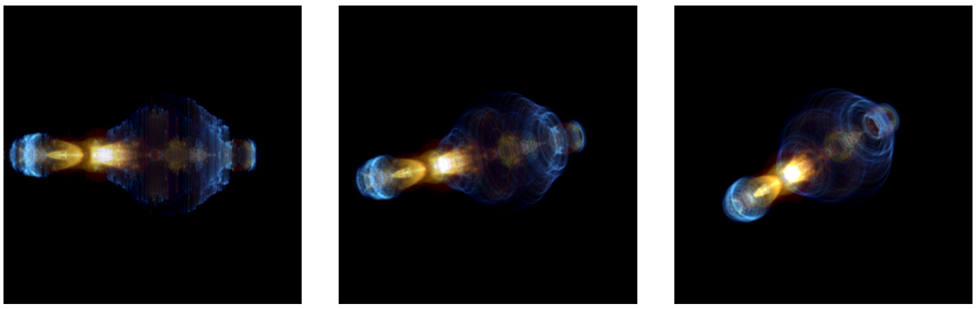


Figure 5.3: Calabash nebula without deformations

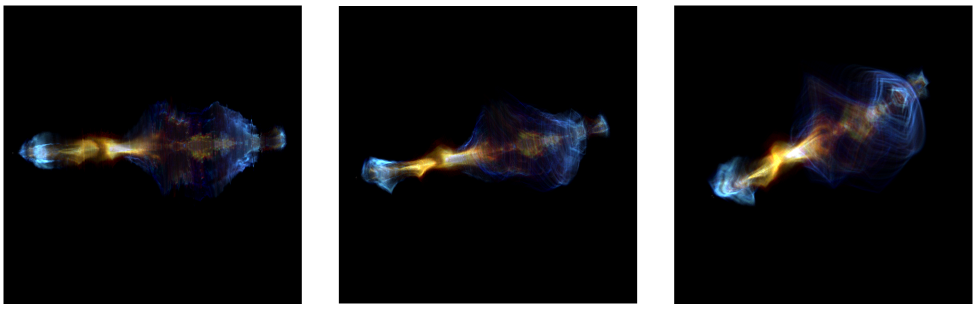


Figure 5.4: Calabash nebula with deformations

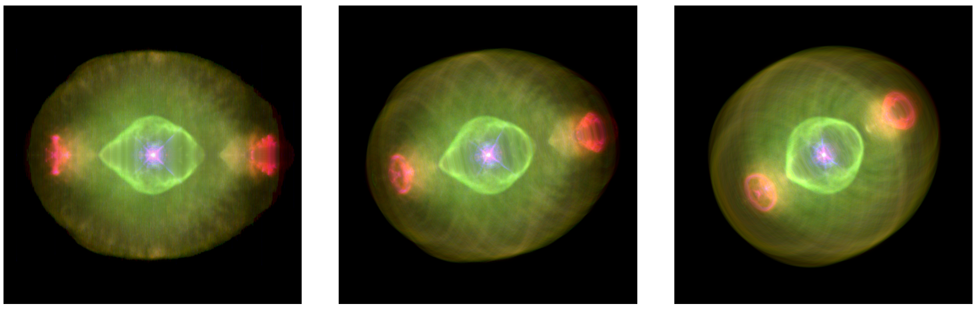


Figure 5.5: NGC 6826 nebula without deformations

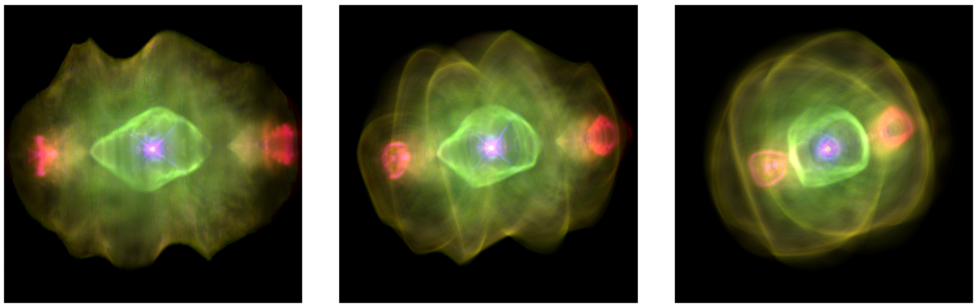


Figure 5.6: NGC 6826 nebula with deformations

5.1.2. Quantitative results

We have evaluated how fast our rendering method can visualize the three nebulae from Section 5.1.1. In addition to that, we have also evaluated how fast our rendering method can visualize a 2D map which is completely empty and a 2D map which is completely emissive. These two maps highlight the cost and/or savings of our optimizations under extreme conditions.

We measure the time that was necessary to render a nebula, both with and without deformations applied. Each nebulae was rendered at an inclination angle of 0°, 15°, 30°, 45°, 60°, 75°and 90°. The nebulae all had a 2D map of size 1024 × 1024 and were rendered to a 2048 × 2048 image. We assumed that the heightmap for each nebulae was already computed. We found that for all nebulae, computing the heightmap of the 2D map was done in under 1 millisecond.

In the tables below, we refer to the “naive” axial-symmetrical volume rendering implementation from Section 3.1.1 as “NASR”. We compared NASR to the different optimizations discussed in Section 3.1. The optimizations shown in the tables below are the *emptiness-check*, *global-maximum-check* and the *ω-step-check* with different values for ω. We used the following values for ω:

- $\omega = \omega_{max} = t_{exit} - t_{entry}$,
- $\omega = \omega_{large} = \frac{1}{32}\omega_{max}$,
- $\omega = \omega_{multi} = \left\{\omega_{max}, \frac{1}{2}\omega_{max}, \frac{1}{4}\omega_{max}, \frac{1}{8}\omega_{max}, \frac{1}{16}\omega_{max}, \frac{1}{32}\omega_{max}\right\}$

ω_{max} is the largest possible step that we could do. It is unlikely that this step can be performed often, but when it is applicable it prevent a significant amount of unnecessary sample points. ω_{large} is a step size which would be applicable more often, but when nebulae contain significant amounts of empty space then we will not traverse the empty space as fast as possible which could cost performance. Finally, ω_{multi} is the hierarchical approach and defines multiple values for ω. We evaluating whether we can perform a large step we start with the largest possible value for ω and try smaller values if the test fails. This approach should ensure that we always skip as much empty space as possible.

In the case of the *global-maximum-check* and *ω-step-check*, we also enable the *emptiness-check*. This means that, in the case that an optimization cannot be applied and a point (or multiple points) can be sampled, we perform the *emptiness-check* before sampling.

All measurements were done on a AMD Ryzen 7 3800X CPU, a Nvdia RTX 2080TI GPU, 64 GB of DDR4 memory and a 1TB SSD.

Table 5.1: Frametime comparisons of the M2-9 nebula

	Without deformations		**With deformations**	
Rendering method	Milliseconds	Compared to NASR	Milliseconds	Compared to NASR
NASR	34,9	1,0	95,3	1,0
emptiness-check	19,0	1,8	17,1	5,6
global-maximum-check	2,1	16,3	3,6	26,7
ω_{max}-step-check	2,3	15,3	3,7	25,7
ω_{large}-step-check	4,1	8,4	5,3	18,0
ω_{multi}-step-check	1,0	34,9	2,7	35,1

Table 5.2: Frametime comparisons of the Calabash nebula

	Without deformations		**With deformations**	
Rendering method	Milliseconds	Compared to NASR	Milliseconds	Compared to NASR
NASR	35,1	1,0	95,0	1,0
emptiness-check	19,4	1,8	18,3	5,2
global-maximum-check	3,7	9,5	5,7	16,6
ω_{max}-step-check	3,9	9,1	5,3	18,0
ω_{large}-step-check	4,6	7,7	6,3	15,1
ω_{multi}-step-check	2,1	16,4	4,1	22,9

Table 5.3: Frametime comparisons of the NGC 6826 nebula

	Without deformations		**With deformations**	
Rendering method	Milliseconds	Compared to NASR	Milliseconds	Compared to NASR
NASR	35,0	1,0	95,9	1,0
emptiness-check	20,4	1,7	24,7	3,9
global-maximum-check	8,3	4,2	17,1	5,6
ω_{max}-step-check	8,7	4,0	17,9	5,4
ω_{large}-step-check	8,7	4,0	16,4	5,8
ω_{multi}-step-check	6,6	5,3	15,3	6,3

Table 5.4: Frametime comparisons of the empty 2D map

	Without deformations		**With deformations**	
Rendering method	Milliseconds	Compared to NASR	Milliseconds	Compared to NASR
NASR	35,1	1,0	95,0	1,0
emptiness-check	19,1	1,8	14,7	6,9
global-maximum-check	< 1	-	< 1	-
ω_{max}-step-check	< 1	-	< 1	-
ω_{large}-step-check	3,6	9,8	3,6	28,3
ω_{multi}-step-check	< 1	-	< 1	-

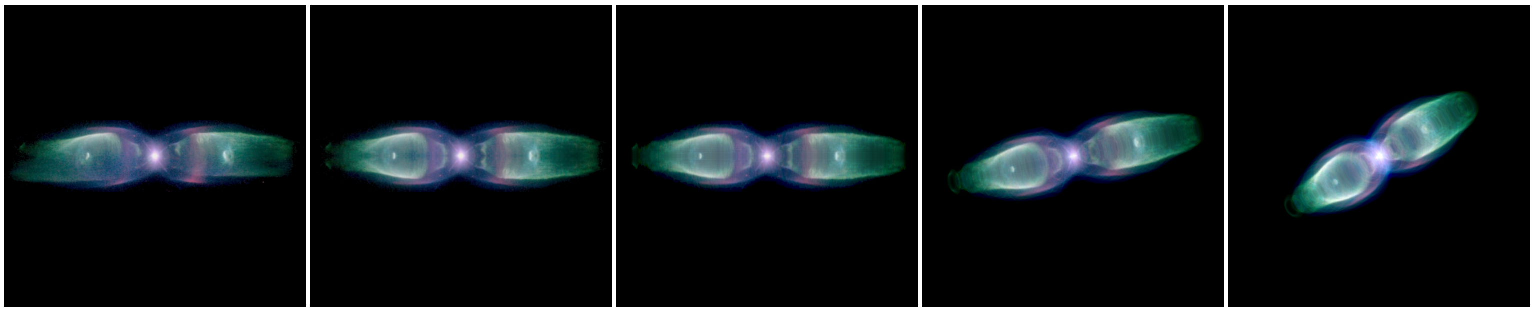
Table 5.5: Frametime comparisons of the emissive 2D map

	Without deformations		**With deformations**	
Rendering method	Milliseconds	Compared to NASR	Milliseconds	Compared to NASR
NASR	32,7	1,0	94,3	1,0
emptiness-check	47,4	0,7	95,7	1,0
global-maximum-check	47,4	0,7	96,9	1,0
ω_{max}-step-check	48,0	0,7	98,0	1,0
ω_{large}-step-check	47,9	0,7	97,6	1,0
ω_{multi}-step-check	48,6	0,7	98,4	1,0

5.2. Reconstruction

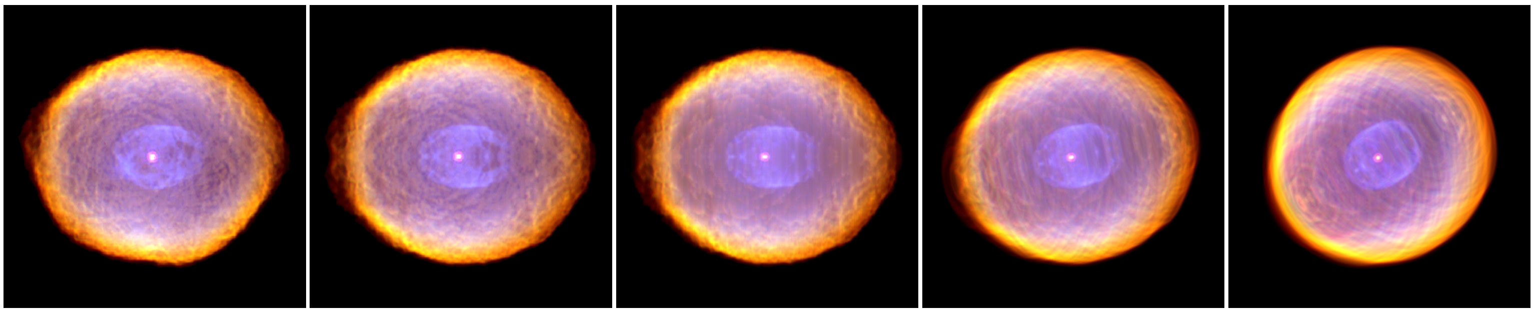
We apply our reconstruction technique on a set of nebula that all have some degree of axial-symmetry in their appearance. We reconstruct a 1024 × 1024 2D map of the nebula and use our rendering method to make images. The position of the symmetry axis in the reference image is assumed to be known before the start of the reconstruction process and any stars in the background of the reference image are removed. The visual results of our method are shown in five figures. Each first figure is the reference image that is used for the reconstruction. Each second figure is a modified version of the reference image and shows what the nebula in the reference image might look like if it were perfectly symmetrical. Each third figure shows the reconstructed nebula when viewed from a viewpoint that makes appear similar to the nebula in the reference image. Each fourth and fifth figure show the reconstructed nebula when viewing it from a novel viewpoint. The time needed for the reconstruction is shown below the figures.

The (visual) quality of our results is similar to that of other methods (Section 2.3.1). Like the results from previous work, our results look plausible. The largest difference between our work and the results of previous work is the running time. Our method runs in the order of minutes, while other methods can take several hours.



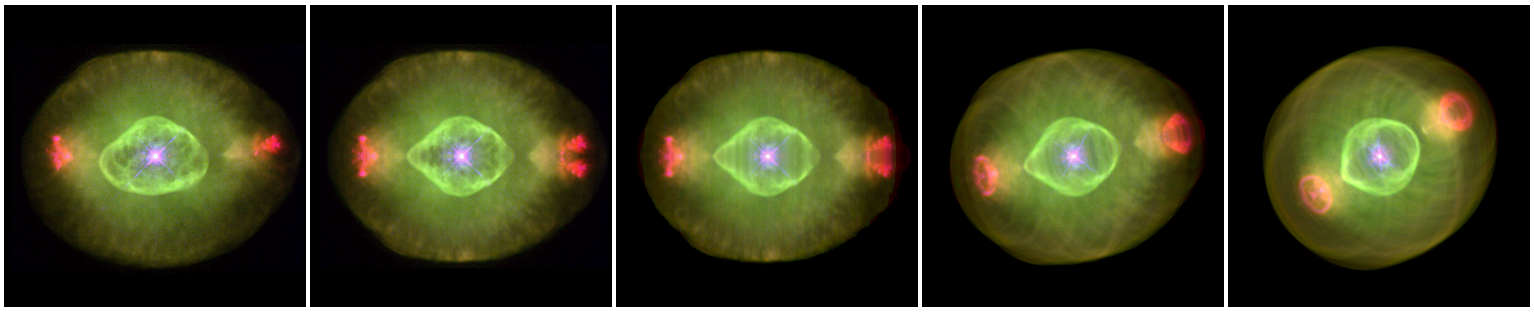
Reference image [4] | Reference with perfect symmetry | Original viewpoint | Novel viewpoint | Novel viewpoint

Figure 5.7: “M2-9” nebula. Reconstruction took 9 seconds.



Reference image [34] | Reference with perfect symmetry | Original viewpoint | Novel viewpoint | Novel viewpoint

Figure 5.8: “IC 418” nebula. Reconstruction took 34 seconds.



Reference image [2] | Reference with perfect symmetry | Original viewpoint | Novel viewpoint | Novel viewpoint

Figure 5.9: “NGC 6826” nebula. Reconstruction took 47 seconds.

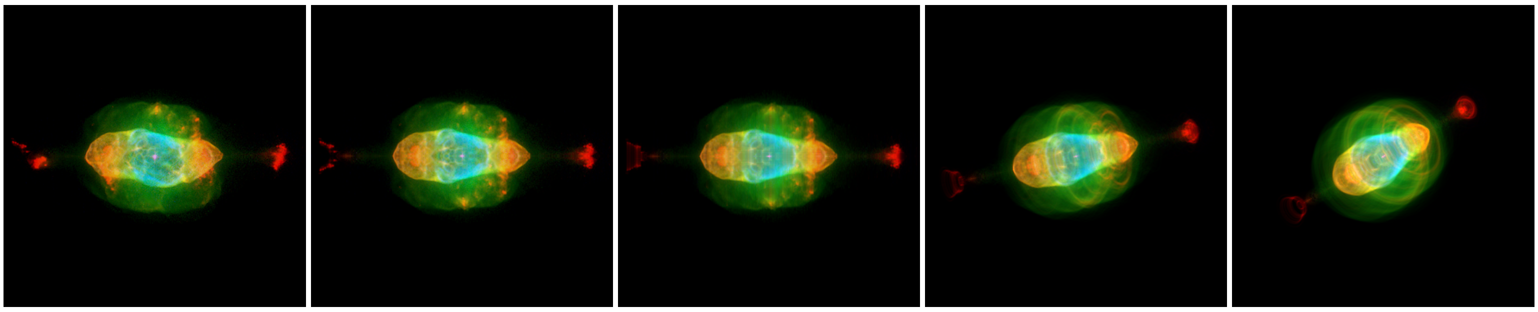


Figure 5.10: “NGC 7009” nebula. Reconstruction took 6 seconds.

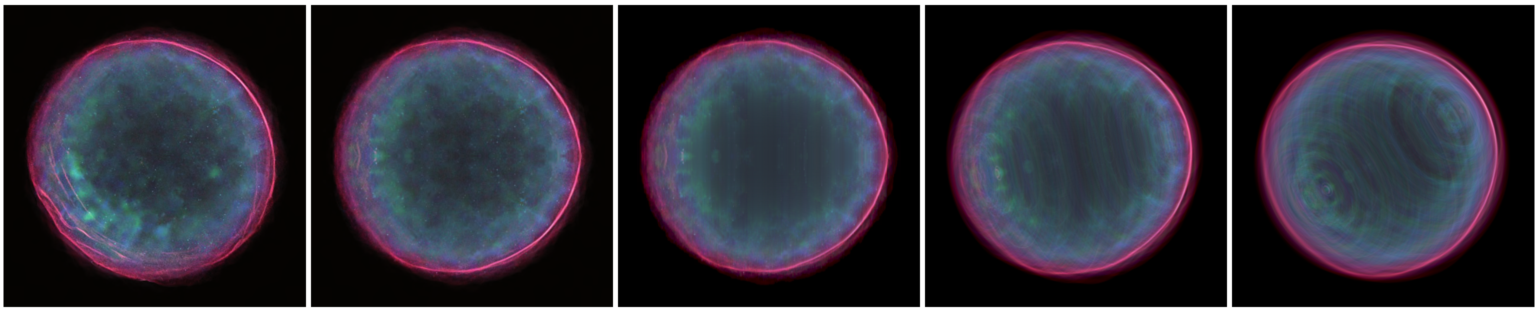


Figure 5.11: “SNR0509” nebula. Reconstruction took 21 seconds.

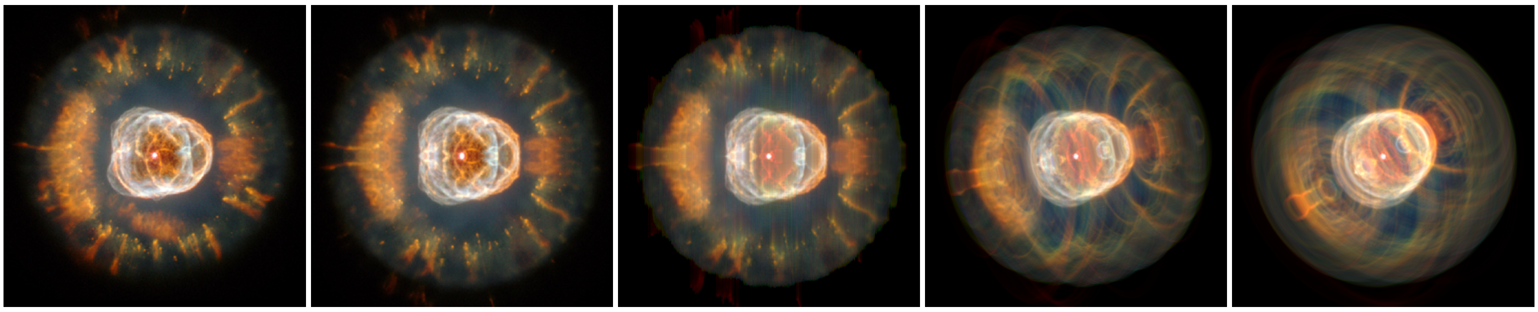


Figure 5.12: “NGC 2392” nebula. Reconstruction took 58 seconds.

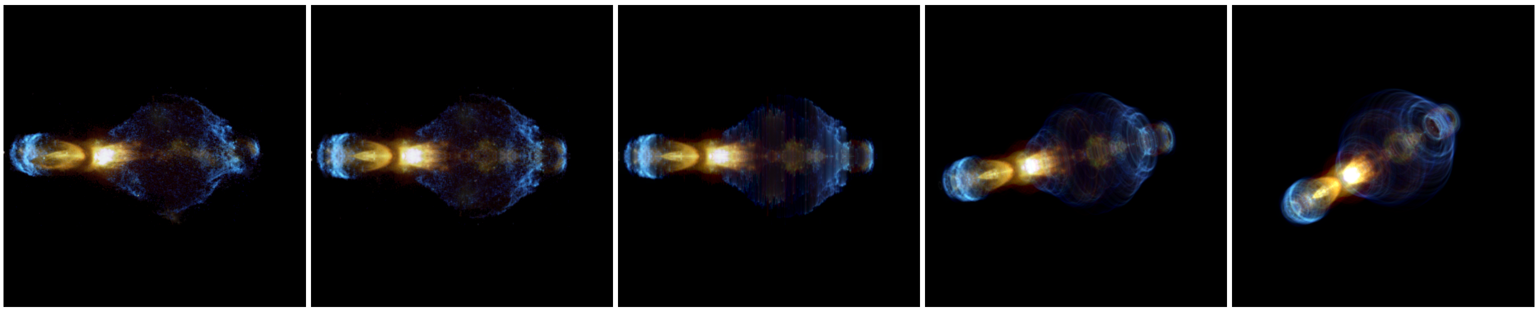


Figure 5.13: “Calabash” nebula. Reconstruction took 13 seconds.

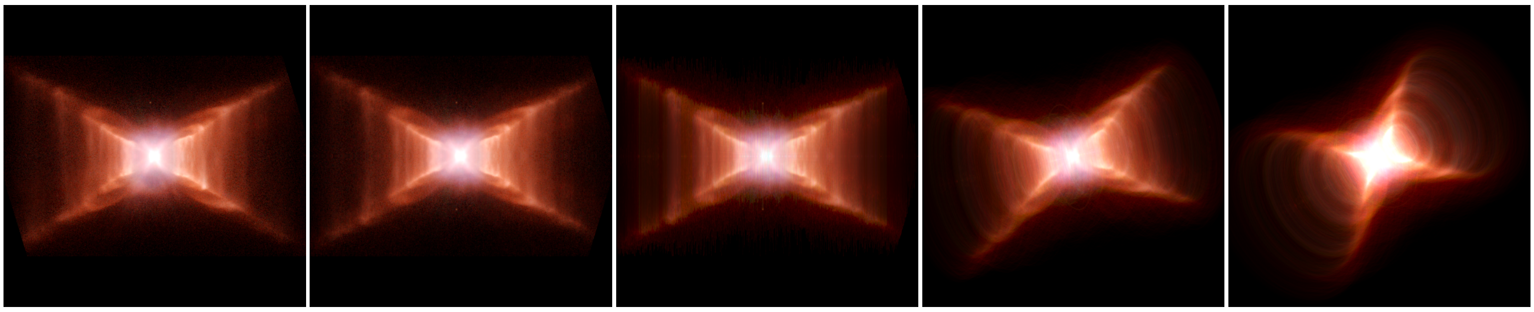

Reference image [32] | Reference with perfect symmetry | Original viewpoint | Novel viewpoint | Novel viewpoint

Figure 5.14: “HD 44179” nebula. Reconstruction took 43 seconds.

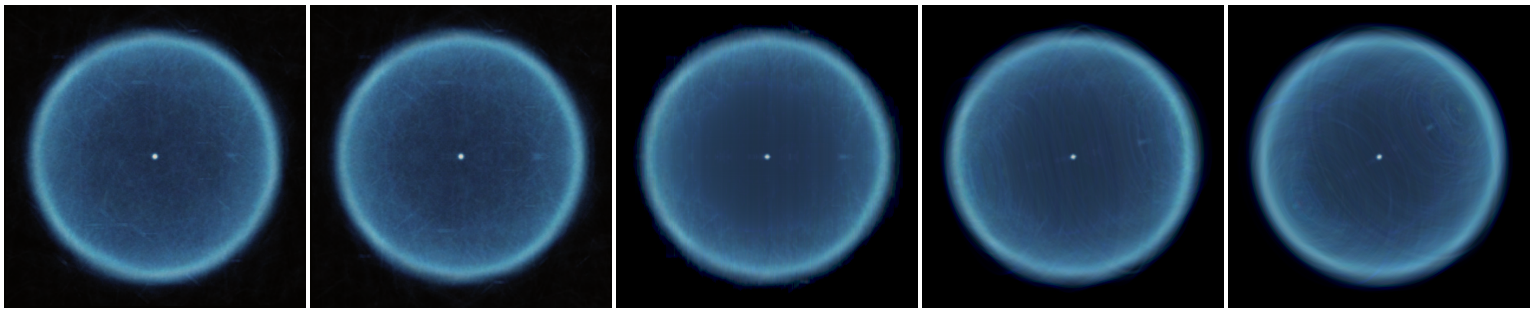

Reference image [48] | Reference with perfect symmetry | Original viewpoint | Novel viewpoint | Novel viewpoint

Figure 5.15: “Abell 039” nebula. Reconstruction took 36 seconds.

6 Discussion

In this Chapter, we will discuss the results from Chapter 5. We will dive deeper into why the results are the way they are and highlight benefits and shortcomings of our visualization and reconstruction methods.

6.1. Visual results of our rendering method

First, we will discuss the visual results that we achieved when we visualize a nebula when no deformations are applied. The nebulae in Figure 5.1, Figure 5.3 and Figure 5.5 look plausible, but simple. Viewing these nebulae at an angle also gives plausible results while the details of the nebula remain visible.

Applying deformations to (re)introduce asymmetry in the appearance of a nebula leads to nebulae that appear more realistic (Figure 5.2, Figure 5.4 and Figure 5.6). We found that the biggest effect was caused by the shape deformations, which introduce irregularities that are also found in real nebula. A variety of color and shape deformations can be applied by the different smooth noise functions that are available. Choosing the right color and shape deformation is a matter of taste but we found that the best results were deformations that accentuated irregularities that were already present in the original nebula.

Renditions of nebulae start appearing artificial when viewing them at a high inclination angle (more than 80°). The nebulae start appearing "flat" when viewing from the side. Perceiving detail about the structure of the nebula is difficult from this angle. Point symmetrical nebulae suffer less from this because they appear the same from any angle. Applying deformations does make the side view of nebulae appear less artificial, but sometimes the result can still lack the detail compared to when viewing the nebula at a less extreme inclination angle.

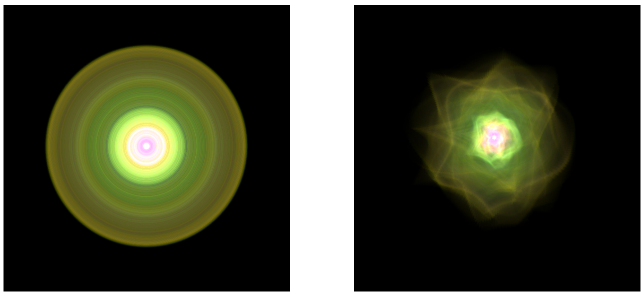


Figure 6.1: Lack of detail when viewing a nebula from the side. Applying deformations mitigates this somewhat.

Many of the details that appear near the symmetry axis (most importantly the center star found in many nebulae) remain visible even when applying our shape deformation technique. This is because sample points that are near the symmetry axis are pulled inwards less than sample points further away. With this in place, the results appear more realistic. Comparing Figure 6.2, it is clear that a loss in details would detract from the realistic appearance of the nebulae.

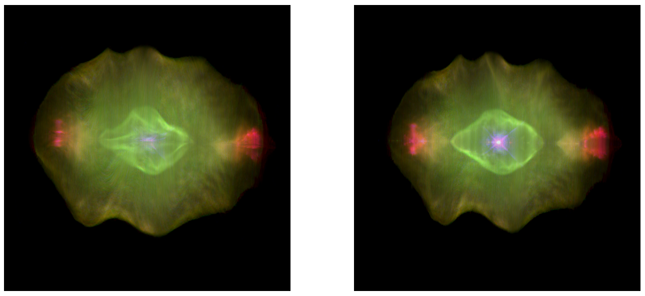


Figure 6.2: Comparison between shape deformation without and with considering the importance of the volume lying close to the symmetry axis.

6.2. Performance of our rendering method

Synthetic 2D maps The results from Table 5.4 and Table 5.5 are to be expected given the workings of our rendering method (and the optimizations that we apply). All our optimizations try to exploit emptiness in a 2D map. As such, all optimizations outperform NASR when tested on the empty 2D map. In the case of the *global-maximum-check*, ω_{max}*-step-check* and ω_{multi}*-step-check*, the speedup is maximal. The volume is immediately traversed from t_{entry} to t_{exit}; the 2D map is not accessed. The performance of ω_{large}*-step-check* is also not unexpected. In that case, we do not test to see if we can traverse the volume in one step but in steps that are still larger than normal. The performance increase therefor is not as high as with the other two ω*-step-check* variants, but is still significant if compared to the NASR or the *emptiness-check*. The performance of the *emptiness-check* compared to NASR proves that it is worth it to prevent accessing the 2D map. Compared to the 2D map, the heightmap is smaller in memory size and can therefor be cached more easily which in turn makes it faster to access. In addition to that, we do not have to do any radiance accumulation but those operations are not extremely costly anyway.

At the opposite end, NASR outperforms all optimizations when visualizing the emissive 2D map, but it does so with a fairly small margin. The slight difference can be explained by the fact that both the heightmap as well as the 2D map have to be accessed, while in the case of NASR, only the 2D map has to be accessed.

Realistic 2D maps The performance results in Table 5.1, Table 5.2 and Table 5.3 highlight the significant performance improvements that we make compared to NASR when visualizing a real nebula. We see that applying deformations costs performance and preventing unnecessary samples is then even more beneficial. With all nebulae, we see that performing the *emptiness-check* already helps with performance. But eliminating multiple sample steps at once is even better for performance. The *global-maximum-check* and ω*-step-check* do precisely this.

The shape of the nebula has a big influence on how well an optimization can be applied. The M2-9 nebula (Table 5.1) is "thin" with lots of empty space surrounding it. It is surrounded by so much empty space that applying the ω_{large}*-step-check* leads to a degradation is performance. The reason for this the same reason why the optimization performed the way it did when visualizing the empty 2D map.

If we look at the performance of the optimizations on the Calabash nebula (Table 5.2), we can, again, conclude that there is enough empty space surrounding the nebula that applying the ω_{large}*-step-check* leads to a degradation in performance. But the lopsided shape of the Calabash nebula also shows that it is inefficient to use the global maximum of the heightmap. Since one side of the nebula is higher than the other, a large amount of rays will go through empty space whilst also going below the global maximum. The ω_{max}*-step-check* uses a local maximum instead and if we look in Table 5.2 then we see that the ω_{max}*-step-check* outperforms the *global-maximum-check* optimization.

If we look at all three cases, we see that the best performing optimization is the ω_{multi}*-step-check*. This is because it will always try to perform as large of a step as possible. Thus, we discard an entire ray when this is possible and otherwise try to skip ahead as much as possible. It leads to the most empty space to be skipped and, with or without deformations, gives a significant speed-up compared to NASR.

6.3. Visual results of our reconstruction method

In our opinion, the reconstruction results of Section 5.2 all look plausible and aesthetically pleasing. All reconstructions look practically identical to (a symmetrical version of) the reference image. The time needed to perform a reconstruction at a level of detail that captures the appearance of the nebula well is also manageable. Especially when compared to the reconstruction times of other methods (Section 2.3), which often could take days. These reconstruction methods are dated and they would run faster on modern hardware, but due to our simplifications to the reconstruction problem, it is expected that these methods are still outperformed.

Since we do not attempt to reconstruct shape and color deformations that cause asymmetry, the more symmetrical the reference nebula is, the better the reconstruction. Examples of this are the nebulae in Figure 5.8, Figure 5.11, Figure 5.14 and Figure 5.15. The difference between the reference image and the symmetric version of the reference image is already fairly small. Viewing these nebula from a novel viewpoint looks very plausible. In the case of Figure 5.11 and Figure 5.15, we see that these nebulae are so spherical that, when viewing them from a novel viewpoint, they look almost exactly the same as when viewing them from the original viewpoint, which can only be explained because of their shape.

We can also see that the reconstruction method has no problem with reconstructing details that lies inside the nebula. For example, the reference nebulae in Figure 5.7, Figure 5.8 and Figure 5.9 all consist of multiple “shells” with outside layers surrounding a more detailed core. The resolution of the 2D map that is used is sufficient to capture these details.

The largest difference between the (symmetric) reference image and reconstructed image can be seen in Figure 5.9 and Figure 5.12 near the center of these nebulae. In the reconstruction, they appear lighter in these areas than in the reference image. This can have two explanations. The first is that in the real nebula, these regions contain absorbent material. Our reconstruction method does not reconstruct absorption and so we cannot replicate its effect. The second explanation is that these nebulae are “thinner” than we assume them to be. We now assume that the nebulae have a constant thickness, but it could be that the nebulae are thinner (so when viewed from the side, the nebulae would be shaped like an ellipse instead of a circle). If they are thinner than not as much radiance can be accumulated and the nebulae will not appear as bright (in the center regions).

The performance of our reconstruction method is sufficiently high that any axial-symmetrical nebula can be reconstructed in a reasonable amount of time. Reconstructions of nebula that are “thin” are faster. Examples are the nebulae in Figure 5.7 and Figure 2.3. This is due to the optimization, which we defined in Section 3.3, whereby we first check which map entries of the 2D map we can expect to be empty. The nebulae in Figure 5.7 and Figure 2.3 simply have fewer map entries that need to be found and the reconstruction time is lower as a result. Still, the time required to reconstruct “larger” nebulae is still fairly low. The most time consuming reconstruction was of the NGC 2392 nebula (Figure 5.12) and it was reconstructed in under one minute.

Overall our reconstruction method leads to reconstructions that appear realistic, even when viewed from a novel viewpoint. We do not capture asymmetries, but still achieve plausible visual results by simulating them during rendering. We also ignore that the nebula that we reconstruct is possibly inclined. This simplifies the reconstruction task and still leads to plausible visual results. Because we ignore the influence of absorption, there can be small visual errors when the reconstruction is compared to reference images. However, only a very small set of the reconstructed nebulae suffer from this and it does not detract from the realistic appearance that these reconstructions have. The performance of our reconstruction method is also fast enough that a high resolution reconstruction can be made in a few seconds rather than a few hours using our reconstruction technique.

7 Conclusion

We have developed a novel rendering method to visualize axial-symmetrical volumes. This technique was specifically developed to visualize axial-symmetrical nebulae and was inspired by techniques used to reconstruct this type of nebulae. Here the 3D volume of a nebula is represented by a 2D image (known as the nebula's 2D map) that is rotated around the symmetry axis. Compared to the traditional voxel representation, the volume can be represented in less memory and modifications can also be made more easily. Our rendering method is optimized to exploit the emptiness that is often found around real axial-symmetrical nebulae to accelerate the visualization of the 2D maps.

We have built our rendering method into an application with which it is possible to view nebulae from any viewpoint. The application also allows for the creation of new axial-symmetrical nebulae or the modification of exiting ones. In addition to that, we propose methods to reintroduce asymmetry in the appearance of the nebulae using shape and color deformations. These shape and color deformations use smooth noise functions (controlled by the user in our application) to introduce tiny irregularities in the appearance of the nebula that break the perfect symmetry that would otherwise be present and allows for the creation of realistically looking nebulae.

Finally, we build upon the previous work in the field and have developed a novel reconstruction approach which allows for the reconstruction of (emissive) axial-symmetrical nebulae. Compared to previous work, our reconstruction method can reconstruct a nebula's 2D map in under a minute, where for previous methods this process would take several hours.

With all of our work, it is possible to get more insight into the appearance and structure of axial-symmetrical nebulae. Our tools can be used to create illustration material suitable for planetariums and astronomy courses.

Future Work During the development of this thesis, we identified several interesting research problems that could be addressed in the future.

We have only focused on axial-symmetrical nebulae and those can be represented by 2D maps. Nebulae that are point-symmetrical can also be represented using a 2D map, because they are implicitly axial-symmetrical, but the resulting 2D map contains redundant information. Instead, a point-symmetrical nebulae could also be represented by a 1D array of values. Reconstructing a nebula that is assumed to be point-symmetrical should be significantly faster than reconstructing an axial-symmetrical nebulae, because only a single column of pixels in the reference image should have enough information to deduce its structure. Whether this will give aesthetically pleasing results remains to be seen, but applying shape and color deformations in a similar manner to how we have done for axial-symmetrical nebulae will probably give results that also look plausible. Many of the visualization optimizations that we have applied would require little modification since the principle on which they operate (the distance between a sampling point and the point of symmetry) also is applicable for point-symmetrical nebulae.

Another research direction that could be studied further concerns the reconstruction of emissive axial-symmetrical nebulae under the same assumptions that we do. Assuming that the nebulae to be reconstructed is not inclined towards Earth means that each column in the 2D map corresponds to a unique column of pixels in the reference image. Since the nebulae is emissive, the radiance received by a pixel is simply a summation of all radiance that lies on the ray coming from the pixel. The problem can then be formulated as the linear system $Ax = b$ with the ith row of A storing how much the entries in x contribute to the ith pixel in b. This system could be solved by a matrix solver, but the non-negativity constraint on the 2D map values should be taken into consideration. Work by Klehm et al. [15, 16] could be used as a starting point, as it deals with a similar equation system.

Finally, our reconstruction method does not reconstruct shape deformations, which results in reconstructions that are perfectly symmetrical. Using the same model as we do to express shape deformations might make the finding of a suitable shape deformation field an ill-posed problem. In addition to that, the current model assumes that all color channels in the 2D map are deformed equally, which means that reconstructing the deformation must be done for all color channels at once. This makes the reconstruction task more difficult. However, the color channels in the reference image usually correspond to particular (narrow) wavelengths of light and this light is emitted by specific kinds of particles. The chemical composition of these particles is unique and as such they are all uniquely effected by interstellar winds. Thus there is a scientific basis to reconstruct the shape deformation of a particular color channel independently of the shape deformations of the other color channels.

Bibliography

[1] Johanna Beyer, Markus Hadwiger, and Hanspeter Pfister. A Survey of GPU-Based Large-Scale Volume Visualization. In R. Borgo, R. Maciejewski, and I. Viola, editors, *EuroVis - STARs*. The Eurographics Association, 2014. ISBN 978-3-03868-028-4. doi: 10.2312/eurovisstar.20141175.

[2] Bruce Balick, Jason Alexander, Arsen Hajian , Yervant Terzian, Mario Perinotto, Patrizio Patriarchi and NASA. Eye-shaped planetary nebula ngc 6826. `"https://hubblesite.org/contents/media/images/1997/38/574-Image.html?news=true"`, 1997.

[3] Bruce Balick, Jason Alexander, Arsen Hajian, Yervant Terzian, Mario Perinotto, Patrizio Patriarchi, NASA/ESA. Hubble's planetary nebula gallery. a view of ngc 7009. `"https://esahubble.org/images/opo9738g/"`, 1997.

[4] Bruce Balick, Vincent Icke, Garrelt Mellema, and NASA/ESA. Hubble sees supersonic exhaust from nebula. `"https://esahubble.org/images/opo9738a/"`, 1997.

[5] O. Cornut. Dear imgui. `https://github.com/ocornut/imgui`, 2020.

[6] Martin Eisemann, Marcus Magnor, Thorsten Grosch, and Stefan Müller. Fast ray/axis-aligned bounding box overlap tests using ray slopes. *Journal of Graphics Tools*, 12(4):35–46, 2007. doi: 10.1080/2151237X.2007.10129248. URL `https://doi.org/10.1080/2151237X.2007.10129248`.

[7] ESA. The eagle nebula. `"https://www.eso.org/public/images/eso0926a/"`, 2009.

[8] ESA Valentin Bujarrabal. `"https://www.esa.int/ESA_Multimedia/Images/2001/08/Calabash_Nebula_Rotten_Egg_Nebula"`, 2000.

[9] E. I. Gislason. Radiative transfer in reflection nebulae. Master's thesis, Technical University of Denmark, DTU Informatics, E-mail: reception@imm.dtu.dk, Asmussens Alle, Building 305, DK-2800 Kgs. Lyngby, Denmark, 2010. Supervised by Assistant Professor Jeppe R. Frisvad, jrf@imm.dtu.dk, DTU Informatics.

[10] Khronos Group. Opencl, 2020. URL `https://www.khronos.org/opencl/`.

[11] Khronos Group. Opengl, 2020. URL `https://www.khronos.org/opengl/`.

[12] Kristian Hildebrand, Marcus Magnor, and Bernd Fröhlich. 3d reconstruction and visualization of spiral galaxies. *Journal of WSCG*, pages 113–120, Jan 2006.

[13] ISO. *ISO/IEC 14882:2017 Information technology — Programming languages — C++*. Fifth edition, December 2017. URL `https://www.iso.org/standard/68564.html`.

[14] Timothy L. Kay and James T. Kajiya. Ray tracing complex scenes. *SIGGRAPH Comput. Graph.*, 20(4): 269–278, August 1986. ISSN 0097-8930. doi: 10.1145/15886.15916. URL `https://doi.org/10.1145/15886.15916`.

[15] Oliver Klehm, Ivo Ihrke, Hans-Peter Seidel, and Elmar Eisemann. Volume stylizer: tomography-based volume painting. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 161–167. ACM Press, 2013. URL `http://graphics.tudelft.nl/Publications-new/2013/KISE13`.

[16] Oliver Klehm, Ivo Ihrke, Hans-Peter Seidel, and Elmar Eisemann. Property and lighting manipulations for static volume stylization using a painting metaphor. *IEEE Transactions on Visualization and Computer Graphics*, 2014. URL `http://graphics.tudelft.nl/Publications-new/2014/KISE14`.

[17] A. Knoll. A survey of octree volume rendering methods. 2006.

[18] Sun Kwok. The origin evolution of planetary nebulae. *The Origin and Evolution of Planetary Nebulae, by Sun Kwok, Cambridge, UK: Cambridge University Press, 2007*, 33, 08 2007. doi: 10.1063/1.1420556.

[19] Ares Lagae, Sylvain Lefebvre, Rob Cook, T. Derose, George Drettakis, David Ebert, J.P. Lewis, Ken Perlin, and Matthias Zwicker. A survey of procedural noise functions. *Computer Graphics Forum*, 29, 12 2010. doi: 10.1111/j.1467-8659.2010.01827.x.

[20] Andrei Lint, Lars Hoffmann, Marcus Magnor, Hendrik P A Lensch, and Hans-peter Seidel. 3D Reconstruction of Reflection Nebulae from a Single Image. *Proceedings of Vision, Modeling, and Visualization*, d:109–116, 2007.

[21] Andrei Lint, Hendrik P A Lensch, and Marcus Magnor. 3D Reconstruction of Emission and Absorption in Planetary Nebulae. (September), 2007.

[22] M.a. Magnor, K. Hildebrand, A. Lintu, and A.J. Hanson. Reflection Nebula Visualization. *IEEE Visualization 2005 - (VIS'05)*, pages 33–33. doi: 10.1109/VIS.2005.86. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1566015`.

[23] Marcus Magnor, Gordon Kindlmann, Charles Hansen, and Neb Duric. Reconstruction and visualization of planetary nebulae. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):485–495, 2005. ISSN 10772626. doi: 10.1109/TVCG.2005.84.

[24] Marcus A. Magnor, Pradeep Sen, Joe Kniss, Edward Angel, and Stephan Wenger. Progress in rendering and modeling for digital planetariums. In Matthew Cooper and Kari Pulli, editors, *31st Annual Conference of the European Association for Computer Graphics, Eurographics 2010 - Areas Papers, Norrköping, Sweden, May 3-7, 2010*, pages 1–8. Eurographics Association, 2010. doi: 10.2312/ega.20101000. URL `https://doi.org/10.2312/ega.20101000`.

[25] Alexander Majercik, Cyril Crassin, Peter Shirley, and Morgan McGuire. A ray-box intersection algorithm and efficient dynamic voxel rendering. *Journal of Computer Graphics Techniques (JCGT)*, 7(3):66–81, September 2018. ISSN 2331-7418. URL `http://jcgt.org/published/0007/03/04/`.

[26] David Nadeau, Jon Genetti, Steve Napear, Bernard Pailthorpe, Carter Emmart, Erik Wesselak, and Dennis Davidson. Visualizing stars and emission nebulas. *Comput. Graph. Forum*, 20:27–33, 03 2001. doi: 10.1111/1467-8659.00472.

[27] NASA and The Hubble Heritage Team (AURA/STScI). Hubble's variable nebula (ngc 2261). `"https://hubblesite.org/contents/media/images/1999/35/904-Image.html"`, 1999.

[28] NASA, ESA and the Hubble Heritage (STScI/AURA)-ESA/Hubble Collaboration. Messier 57 (the ring nebula). `"https://www.nasa.gov/feature/goddard/2017/messier-57-the-ring-nebula/"`, 2017.

[29] NASA, ESA, and the Hubble SM4 ERO Team. `"https://www.nasa.gov/mission_pages/hubble/multimedia/ero/ero_ngc6302.html"`, 2009.

[30] NASA, ESA, Andrew Fruchter, and the ERO team. `"https://esahubble.org/images/heic9910a/"`, 2000.

[31] NASA, ESA, CXC, SAO, the Hubble Heritage Team, and J. Hughes. `"https://hubblesite.org/contents/news-releases/2012/news-2012-06.html"`, 2012.

[32] NASA, ESA, Hans Van Winckel, and Martin Cohen. `"https://hubblesite.org/contents/news-releases/2004/news-2004-11.html"`, 2004.

[33] NASA, ESA, M. Robberto (Space Telescope Science Institute/ESA) and the Hubble Space Telescope Orion Treasury Project Team). Hubble panoramic view of orion nebula reveals thousands of stars. `"https://esahubble.org/news/heic0601/"`, 2006.

[34] NASA/ESA and The Hubble Heritage Team (STScI/AURA). The spirograph nebula. `"https://esahubble.org/images/opo0028a/"`, 2000.

[35] NASA/Gary Stevens, picture prepared by Adrian Pingstone. The witch head nebula. `"https://apod.nasa.gov/apod/ap010227.html"`, 2001.

[36] NASA/Lynn Hilborn. Lynds dark nebula 1251. `"https://apod.nasa.gov/apod/ap160930.html"`, 2016.

[37] Nü. "https://upload.wikimedia.org/wikipedia/commons/3/35/Octree2.png", 2006.

[38] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 10.2.89, 2020. URL https://developer.nvidia.com/cuda-toolkit.

[39] OpenMP Architecture Review Board. OpenMP application program interface version 2.0, 2002. URL https://www.openmp.org/wp-content/uploads/cspec20.pdf.

[40] J. Peck. Fastnoise simd. https://github.com/Auburn/FastNoiseSIMD, 2020.

[41] K. Perlin. Chapter 2 noise hardware.

[42] William H. Press, William T. Vetterling, Saul A. Teukolsky, and Brian P. Flannery. *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, USA, 2nd edition, 2001. ISBN 0521750334.

[43] D. Ruijters and Anna Vilanova. Optimizing gpu volume rendering. In *WSCG - Winter School of Computer Graphics*, volume 14, pages 9–16, Feb 2006. URL http://graphics.tudelft.nl/Publications-new/2006/RV06.

[44] Science: NASA, ESA, C.R. O'Dell (Vanderbilt University), M. Meixner and P. McCullough (STScI); Illustration: NASA, ESA, G. Bacon (STScI). A new twist on an old nebula. "https://hubblesite.org/contents/news-releases/2004/news-2004-32.html", 2004.

[45] Wolfgang Steffen, Nicholas Koning, Stephan Wenger, Christophe Morisset, and Marcus Magnor. Shape: A 3D modeling tool for astrophysics. *IEEE Transactions on Visualization and Computer Graphics*, 17(4): 454–465, 2011. ISSN 10772626. doi: 10.1109/TVCG.2010.62.

[46] Annemieke Verbraeck and Elmar Eisemann. Interactive black-hole visualization. *IEEE Transactions on Visualization and Computer Graphics*, page 10, 2020. URL http://graphics.tudelft.nl/Publications-new/2020/VE20.

[47] Stephan Wenger, Marco Ament, Stefan Guthe, Dirk Lorenz, Andreas Tillmann, Daniel Weiskopf, and Marcus Magnor. Visualization of astronomical nebulae via distributed multi-gpu compressed sensing tomography. *IEEE Transactions on Visualization and Computer Graphics*, 18:2188–2197, 12 2012. doi: 10.1109/TVCG.2012.281.

[48] WIYN/NOIRLab/NSF. "https://noirlab.edu/public/images/noao0102a/", 2002.