Visualization of Astronomical Nebulae via Distributed Multi-GPU Compressed Sensing Tomography

Stephan Wenger, Marco Ament, Stefan Guthe, Dirk Lorenz, Andreas Tillmann, Daniel Weiskopf, *Member, IEEE Computer Society*, and Marcus Magnor, *Member, IEEE*

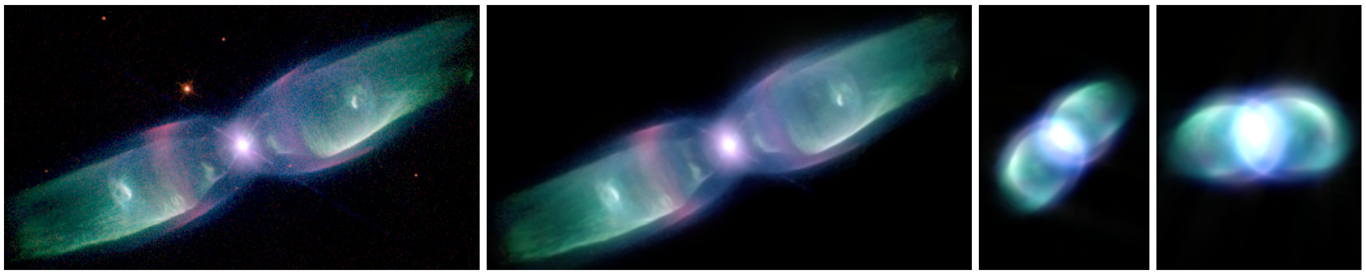

Fig. 1. The planetary nebula M2-9 is a typical example of a bipolar nebula. Its quasi-symmetric twin lobes of ionized material emanate from a binary star system in its center. Assuming axial symmetry, our reconstruction algorithm uses a single input image (a) to produce a high-resolution 3D visualization that closely resembles the original image when rendered from the same viewpoint (b). From a novel vantage point, the emission along the principal axis of the nebula accumulates and creates a luminous halo (c). As the vantage point approaches the symmetry axis, the received intensity further increases and the perceived shape of the nebula changes toward two entangled rings (d). The resolution of the reconstructed volume is 512^3 voxels. *Original image: Bruce Balick (University of Washington), Vincent Icke (Leiden University, The Netherlands), Garrelt Mellema (Stockholm University), and NASA.*

Abstract—The 3D visualization of astronomical nebulae is a challenging problem since only a single 2D projection is observable from our fixed vantage point on Earth. We attempt to generate plausible and realistic looking volumetric visualizations via a tomographic approach that exploits the spherical or axial symmetry prevalent in some relevant types of nebulae. Different types of symmetry can be implemented by using different randomized distributions of virtual cameras. Our approach is based on an iterative compressed sensing reconstruction algorithm that we extend with support for position-dependent volumetric regularization and linear equality constraints. We present a distributed multi-GPU implementation that is capable of reconstructing high-resolution datasets from arbitrary projections. Its robustness and scalability are demonstrated for astronomical imagery from the Hubble Space Telescope. The resulting volumetric data is visualized using direct volume rendering. Compared to previous approaches, our method preserves a much higher amount of detail and visual variety in the 3D visualization, especially for objects with only approximate symmetry.

Index Terms—Astronomical visualization, distributed volume reconstruction, direct volume rendering.

1 Introduction

Due to their intricate and colorful structure, astronomical nebulae are among the most visually appealing astrophysical phenomena. However, our vantage point is confined to the solar system, and we can only gather imagery and other observational data from a single point of view. This makes deducing the correct 3D geometry a notoriously difficult task except for cases when the geometry is particularly simple and additional data from specially equipped telescopes is available [32]. Fortunately, for the purpose of visualization in education and popular science, e.g. in digital full-dome planetariums and sky simulation software such as Celestia (`http://www.shatters.net/celestia/`), a plausible and realistic volumetric reconstruction is often sufficient. Such a plausible but not necessarily physically accurate reconstruction can also give astronomers an initial intuition about possible geometries and may serve as a starting point for further manual modeling. For example, it has been shown only recently using manual modeling that some classes of nebulae that were believed to be structurally different actually might share a common morphology but are only observed from different vantage points [8]. Such structural insight could be obtained directly from a 3D visualization like the one we propose.

3D information can be obtained from a single image by exploiting the fact that many types of astronomical nebulae exhibit an approximate spherical or axial symmetry [19]. Based on the assumption that the object looks very similar from a number of different directions, the view from Earth can be replicated at other virtual viewpoints, resulting in a tomographic reconstruction problem (Section 3). Tomographic reconstruction traditionally uses *filtered back-projection*, which suffers from a number of limitations, including susceptibility to noise and artifacts as well as the lack of flexible regularization schemes to alleviate the derogatory effects of inconsistent data and incomplete sampling. Since the symmetry of astronomical nebulae is only approximate, the virtual views tend to be inconsistent. Thus, a different approach has

- *Stephan Wenger, Stefan Guthe and Marcus Magnor are with the Institut für Computergraphik, TU Braunschweig, Germany. E-mail:* `wenger@cg.cs.tu-bs.de`.
- *Marco Ament and Daniel Weiskopf are with VISUS, University of Stuttgart, Germany.*
- *Dirk Lorenz is with the Institute for Analysis and Algebra, TU Braunschweig, Germany.*
- *Andreas Tillmann is with the Research Group Optimization, TU Darmstadt, Germany.*

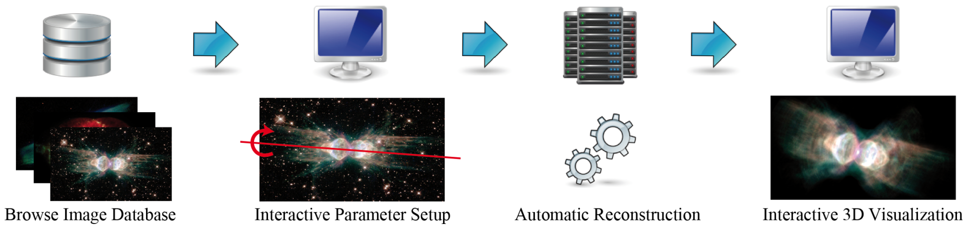

Fig. 2. Overview of our application pipeline. The user browses an image database, such as `http://hubblesite.org/gallery/`, and selects an astronomical object with a roughly symmetric shape. Next, a few relevant parameters are interactively chosen by the user, determining the type of symmetry and the desired volume resolution. Based on this parameter setup, our fully automatic reconstruction algorithm solves a constrained optimization problem on a multi-GPU cluster to compute a volumetric model that closely resembles the view of the original image taken from Earth. The result of our algorithm is a volumetric model of the nebula that can be visualized interactively with direct volume rendering. The visualization is also valuable to improve the overall quality of the result by slightly readjusting the parameters.

to be taken. In the context of computed tomography, a number of *iterative algebraic reconstruction techniques* have been introduced that attempt to reduce the drawbacks of filtered back-projection. These approaches are, in general, more stable with respect to noise or occluders and more flexible with respect to missing data, inconsistent data, or non-equidistant projections. However, iterative techniques require a considerably higher computational effort, and the medical industry has been hesitant to employ them in commercial products [30]. As part of our contribution, we alleviate this limitation by designing and implementing an efficient iterative tomographic reconstruction algorithm for a multi-GPU compute cluster. Thus, the advantages of iterative techniques can be exploited even for the large datasets required for visually appealing renderings.

Our algorithm is based on a particularly promising formulation of the tomographic reconstruction problem that originates from the theory of *compressed sensing* [5,6]. Compressed sensing states that under certain conditions a signal can be perfectly reconstructed from only a small number of measurements. Most importantly, the signal is assumed to be sparse in some transform domain such as a wavelet space, the gradient domain or, as in our case, in a voxel representation. Even if this condition is often not perfectly fulfilled in practice, as is the case in most tomographic applications, the corresponding algorithms are sufficiently robust to yield good approximate solutions in cases where the sparsity condition is only roughly satisfied. Our algorithm builds on the *fast iterative shrinkage-thresholding algorithm* [3]. As part of our contribution, we adapt it to the tomographic reconstruction problem and extend it with additional constraints for enforcing nonnegative intensities and selected projections, see Figure 3. The optimization algorithm is discussed in detail in Section 4.

The computational bottleneck of most iterative reconstruction techniques, including those based on compressed sensing, is the repeated computation of forward and backward *projections* (Section 5). The *forward* projection denotes the projection of a discretized volume into different views, which can in many cases be described as a linear operator. The *backward* projection is, in a mathematical sense, the adjoint of the forward projection operator and projects the views back into the volume. In our implementation, both projection operators are distributed among the GPUs in a compute cluster to speed up this most computationally expensive operation (Section 6). Since the forward and backward projection operations are common to most iterative reconstruction techniques, this core part of our parallel implementation can be integrated in a wide range of other algorithms as well.

The application of our approach results in a simple workflow, see Figure 2, as opposed to cumbersome manual modeling. An astronomical image database serves as an input archive with high-resolution images taken from modern telescopes, such as the Hubble Space Telescope. The user selects an image of an astronomical nebula with approximately spherical or axial symmetry and creates a simple setup for the subsequent fully automatic reconstruction. As part of the setup, the user specifies the type of symmetry (along with the symmetry axis, if applicable), the desired resolution of the resulting volume, and optional parameters. Afterwards, our reconstruction algorithm solves a constrained optimization problem and computes a volumetric model of the nebula that can be visualized interactively with standard direct volume rendering.

A fundamental advantage of our application compared to conventional modeling tools is the small number of parameters and their simple handling. The number of virtual viewpoints controls the smoothness of the reconstruction. The spatial arrangement of the virtual viewpoints is basically determined by the symmetry of the nebula with a single additional parameter for jittering the center of symmetry. One of our contributions to the core optimization algorithm is to constrain the view from Earth to be similar to the original image; the level of similarity is controlled by a single parameter that determines the number of inner loops in the algorithm. The last parameter controls the magnitude of regularization, which is necessary as the reconstruction problem is usually ill-posed. We will discuss and evaluate typical choices of parameters along with the results displayed in Section 7.

2 Related Work

Because of the difficulty of deducing plausible three-dimensional structure from a single image, the reconstruction of volumetric models for astronomical nebulae is typically performed manually by astronomers or artists. For example, a complex 3D model of the Orion nebula was created by professional astronomers over several years [28,38]. Even with specialized modeling tools [34], the typical modeling time is still measured in weeks. Attempts on symmetry-based automatic reconstruction and visualization [16,19,20] have produced perfectly symmetric, low-resolution models poor in visual detail. Automated methods for the reconstruction and visualization of asymmetric reflection nebulae [18] suffer from a similar lack of detail and are not applicable for translucent objects, such as most planetary nebulae. Although it is theoretically possible to introduce artificial asymmetry and detail into the reconstruction results [36], this process is as complex as the original reconstruction problem and often results in unappealing visual artifacts like streaks and implausible clusters of emission.

Tomographic reconstruction is common in medical applications such as computed tomography, but can also be used for other transparent volumetric phenomena like flames [12]. Iterative reconstruction techniques for computed tomography date back to the 1970s [10]. Since then, numerous iterative reconstruction algorithms have been proposed, including some based on compressed sensing. A comprehensive overview of iterative algorithms based on minimization of the total variation can be found in [35]. However, it becomes clear from the following argument that these algorithms are, in general, not suited for large-scale volumetric reconstruction problems with arbitrary projection geometries.

The memory requirements for fully volumetric reconstruction algorithms quickly become unmanageable. For example, a 1024^3 32-bit floating-point voxel volume requires 4 GB of memory. For optimal results, the number of input images should be of the same order of magnitude as the resolution of the volume, such that the intermediate images require about 4 GB as well. Both amounts are doubled by the

```
$L \leftarrow 2$ times the largest eigenvalue of $A^{\mathrm{T}}A$
$x_0 \leftarrow \max\left(\frac{A^{\mathrm{T}}b-\tau}{L}, 0\right)$
$y_0 \leftarrow x_0$
$t_0 \leftarrow 1$
for $i = 1$ to $n_{\mathrm{outer}}$ do
  $x_i \leftarrow \max\left(\frac{\frac{L^2}{2}y_i - LA^{\mathrm{T}}(Ay_i-b)-\tau}{\frac{L^2}{2}}, 0\right)$
  for $j = 1$ to $n_{\mathrm{inner}}$ do {optional loop to enforce $Bx = c$}
    $x_i \leftarrow x_i + B^{\mathrm{T}}(c - Bx_i)$
    $x_i \leftarrow \max\left(\frac{\frac{L^2}{2}y_i - LA^{\mathrm{T}}(Ay_i-b)-\tau}{\frac{L^2}{2}}, 0\right)$
  end for
  $t_i \leftarrow \frac{1}{2}\left(1+\sqrt{1+4t_{i-1}^2}\right)$
  $y_i \leftarrow x_i + \frac{(t_{i-1}-1)}{t_i}(x_i - x_{i-1})$
end for
return $x_{n_{\mathrm{outer}}}$
```

Fig. 3. Optimization algorithm derived from [3]. Given linear operators A and A^{T}, a data vector b, the regularization parameter τ, and the iteration limit n_{outer}, it computes the vector $x \geq 0$ that minimizes $\|Ax-b\|^2 + \tau\|x\|_1$. Optionally, a constraint of the form $Bx = c$ can be specified by passing linear operators B and B^{T} (with $BB^{\mathrm{T}} = \mathbb{I}$), a constraint vector c, and the iteration limit n_{inner}. The operators A, A^{T}, B, and B^{T} only need to be given implicitly, i.e., as functions that compute the application of the operator to a vector. In the case of volumetric projection operators, this allows for a much more efficient implementation than an explicit matrix multiplication; in fact, explicit storage of the matrix elements quickly becomes infeasible. The constant L can be computed from implicitly given A and A^{T} using a power iteration scheme. The results of applying A and A^{T} can be cached, while Ay can be computed from Ax, so that only one evaluation of the operators A and A^{T} is necessary in each step.

fact that the variables of the previous iteration are often required in the update step. Furthermore the original input images add another 4 GB, leading to a total of 20 GB of memory. For many tomographic applications based on optical imagery, the images contain RGB color information, raising the amount of memory required to about 60 GB. Because of the complexity of the generic volumetric reconstruction problem, many reconstruction algorithms are tailored to specific projection setups [14, 33, 37], limiting their range of applicability. Most importantly, the arbitrary random projection geometries required for the reconstruction of spherically symmetric nebulae disallow any such optimization.

In order to solve the generic volumetric reconstruction problem efficiently for large-scale datasets, the use of massively parallel computation, e.g. on graphics hardware, is indispensable. Unfortunately, the memory of current GPUs is far smaller than what is required for high-resolution reconstructions. This requires either costly swapping between GPU memory and CPU memory or disk, or the use of multiple GPUs whose memory and processing power are effectively combined. The only reconstruction algorithm known to the authors that makes use of multiple GPUs [13], however, is limited to a single machine with typically not more than two or four graphics cards. In addition, it neither offers any means for additional constraints, nor does it provide for any kind of regularization. Instead, it computes a simple least-squares approximation of the data, making it unsuitable for the plausible reconstruction of highly inconsistent datasets. In contrast, the proposed approach is flexible enough to provide hard constraints combined with different regularization schemes and makes use of a virtually unlimited amount of GPU memory by distributing the computational load to multiple computers in a cluster. As iterative reconstruction techniques typically scale at least polynomially with problem size, the additional speedup due to parallel computation on a large number of devices is an important additional feature of our approach that effectively reduces computation time from several days to a few hours.

In direct volume rendering, distributed algorithms and GPU acceleration are widely used to improve data and performance scalability. The two fundamental paradigms [25] in parallel volume rendering decompose either image-space (sort-first) or object-space (sort-last) to distribute data and workload among the render nodes. In addition, GPUs can be employed to further accelerate rendering on each node, for example in sort-last approach with load balancing [27] or with multiple GPUs in a single physical node [21]. Typically, sort-last decompositions scale well with large data, but have to deal with more advanced compositing techniques such as binary swap [17], direct send [7], or radix-k [31]. More recently, distributed sort-first volume rendering has been employed [26] in conjunction with GPU acceleration and load balancing, avoiding a final compositing step but sacrificing the ability to scale easily with large data. In contrast to these techniques, volumetric reconstruction requires images from more than one view point and also performs backward projections. Together, this leads to a much higher computational expense and requires different strategies in a distributed environment than previous methods provide.

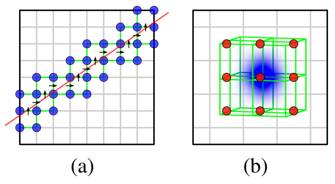

Fig. 4. Traversal for a single output pixel (viewing ray drawn in red) through the volume data during forward projection with all contributing voxels marked blue (a), compared against the support region of a single voxel (blue) in the backward projection with all contributing pixels marked red (b). In both illustrations, the affected grid cells of the volume are indicated in green.

3 Image Formation and Symmetry

Our method is based on two fundamental assumptions: a model of emissive light transport within astronomical nebulae and an assumption about their spatial symmetry, which we will introduce in the following.

In general, in tomographic applications, an object is imaged from several different views, from which a discretized volumetric representation of the object can be reconstructed. The imaging process consists of projecting the volume to these views according to an optical model of emission and absorption [23]. Many astronomical objects like planetary nebulae, see Figure 1, exhibit little to no absorption. By neglecting any effects of absorption, the image formation can be described by a linear system of equations. The intensity $I = \sum_i e_i d_i$ of an image pixel is then a linear combination of the emission densities $e_0, e_1, \ldots, e_n$ along the corresponding viewing ray, where d_i is the length of the viewing ray segment that falls into the i^{th} volumetric grid cell along the ray. When the emission densities of the volume are written as a vector v of grid cell intensities and the intensities of the pixels in the k^{th} view are written as a vector b_k of pixel values, we can define the forward projection as a linear operator M_k such that $M_k v = b_k$. The transpose of the forward projection operator, the backward projection operator M_k^{T}, is equally important for the mathematical formulation and the implementation of the algorithm. Intuitively, it distributes the intensity of each pixel among all contributing grid cells proportionally to their contribution. In Section 5, we will discuss practical considerations for efficient and accurate implementation of the forward and backward projection operators.

In the context of tomographic reconstruction, assumptions about spatial symmetry can conveniently be modeled by reconstructing the volume from a number of *virtual* views. The viewpoint and image content of these views define the type of symmetry. For example, a spherically symmetric object looks the same from every possible viewpoint; this can be modeled by creating a number of *random* viewpoints and associating a copy of the original observed image with each viewpoint. With respect to enforcing exact symmetry using an analytical model, this approach has the advantage of allowing small deviations that create more variety in the visualization and are important for being able to discern different views of the 3D object. A larger number of virtual

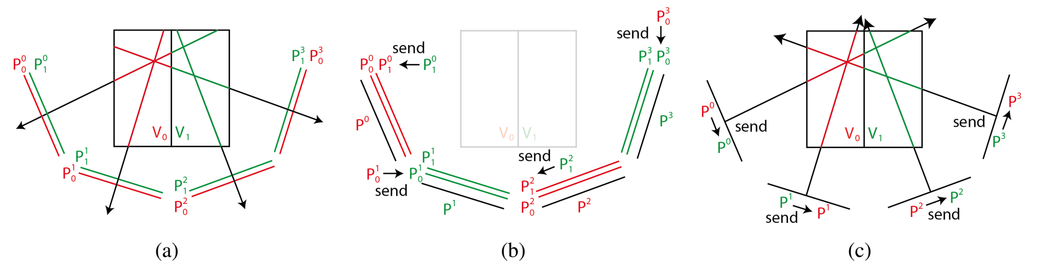

Fig. 5. Overview of our distributed projection steps with two compute nodes (red and green) and four projected views. The partial volume datasets V_0 and V_1 are distributed evenly across both nodes. For the forward projection, each node i applies the linear operator A to its sub-volume V_i with a forward projection k on its GPU and obtains a set of partial images P_i^k (a). The compositing workload of the partial images of all projected views is distributed evenly among all nodes. Therefore, a subset of the partial images is transferred over the network and all nodes perform the computation concurrently on their GPU to obtain P^k (b). For the backward projection, the composited images are spread to all nodes. Afterward, the inverse operator A^T is applied to the images P^k (c).

views create more accurate symmetry, whereas a smaller number introduce more variety and a more realistic 3D impression. In addition, the concept of random virtual views flexibly adapts to other types of symmetry. For example, axially symmetric objects can be modeled by arranging the virtual views around the axis of symmetry. By randomly perturbing the axis for each individual view, additional variance can be introduced to aid the perception of depth. Examples of both types of symmetry are presented in Section 7.

4 Compressed Sensing Algorithm

In the previous section, it has been shown that the projection of a volume v to an image b_k can be written as a linear equation $M_k v = b_k$. In a typical tomographic application, many (say n_views) images b_k will be captured. By stacking these image vectors and the corresponding operators M_k, we can summarize the complete capturing process in a system of linear equations

$$(M_0, \ldots, M_{n_\text{views}-1})^\mathrm{T} v = (b_0, \ldots, b_{n_\text{views}-1})^\mathrm{T} \quad \text{or} \quad Mv = b\,. \tag{1}$$

When the projections M and the captured images b are known, the volumetric object v can in principle be reconstructed by solving this system of linear equations. However, in practice, this inverse problem is often ill-posed. For example, there is often not enough information captured in the images b to uniquely define the volume v. In this case, the most plausible solution has to be selected by choosing an appropriate *regularizer*.

In the context of compressed sensing, it is typically assumed that natural signals are *sparse* in some transform domain. For example, a photograph of an outdoor scene may be sparse in a wavelet representation, which is why such representations are used for image compression. Compressed sensing algorithms promote such sparse solutions by solving an optimization problem that includes a regularization term of the form $\|x\|_1 = \sum_i |x|$, where $x = Sv$ is the signal vector v transformed to some sparsity domain S. Equation 1 can thus be written as $Ax = b$ with $A = MS^{-1}$ and $x = Sv$.

In general, any linear basis transform S can be used as a sparsity basis. In a typical tomography application where an isolated object is imaged in front of a dark background, the signal can be assumed to be sparse in the voxel representation, so that S is the identity. In our numerical experiments, we limit ourselves to such voxel-domain sparsity for the sake of simplicity. Nevertheless, other sparsity domains can be used by simply choosing a different S.

The fast iterative shrinkage-thresholding algorithm [3] is an example of a fast, state-of-the art compressed sensing signal recovery algorithm. It iteratively minimizes the generic functional $F(x) = f(x) + g(x)$ for convex $g(x)$ and convex continuously differentiable $f(x)$. Part of our contribution consists in adapting this algorithm to the tomographic reconstruction problem and extending it with an option to enforce nonnegativity of intensities as well as additional constraints. We choose f as a *data fidelity term* $f(x) = \|Ax - b\|^2$ that enforces compliance of the solution with the captured images, and g as a *regularization term* $g(x) = \tau \|x\|_1 = \tau \sum_i \|x_i\|$ (with the ℓ_1 norm of x scaled by a weighting factor τ) that promotes sparse solutions. In simple terms, minimization of the ℓ_1 norm does not encourage a uniform distribution of intensity as a least-squares method would do, but instead penalizes nonzero coefficients. For a more comprehensive introduction, we refer to Baraniuk [2].

Two other popular choices for regularization terms include the ℓ_1 norm of the wavelet coefficients of x and the ℓ_1 norm of the gradient of x, or total variation (TV). We choose to minimize the ℓ_1 norm of x because it integrates more easily with the requirement of nonnegative intensities than a minimization of the wavelet coefficients, which often leads to overshooting and ringing artifacts. Compared to minimization of the total variation, our approach is computationally much more efficient, and it preserves fine detail that is easily suppressed by TV regularization. Most importantly, minimizing the voxel intensities creates compact objects on a clear low-intensity background, which is a favorable property for the reconstruction of isolated astronomical objects.

The complete optimization process is depicted in Figure 3. In an initialization step, the smallest Lipschitz constant L of ∇f is computed from the largest eigenvalue of $A^\mathrm{T}A$. Due to memory constraints (see Section 5), the forward and backward projection operators A and A^T are only given implicitly, i.e., matrix products Ax and $A^\mathrm{T}x$ can be computed, but the individual matrix elements are unknown. Therefore, the largest eigenvalue is computed using the power iteration method [9, p. 330]. Subsequently, $F(x)$ is iteratively optimized. The number n_outer of iteration steps specifies the tradeoff between runtime and reconstruction quality. Alternatively, a threshold for the change in x or $F(x)$ can be used as a termination criterion. In each iteration i,

$$\frac{L}{2}\left\| x_i - \left(y_i - \frac{1}{L}\nabla f(y_i)\right)\right\|^2 + g(x_i) \tag{2}$$

is minimized, where y_i is derived from the vectors x_i and x_{i-1} of the previous steps as described in Figure 3. Since all density values are inherently nonnegative, we additionaly require $x_i \geq 0$. With our choices for f and g, Equation 2 becomes

$$\left\| \frac{L}{2}(x_i - y_i) + A^\mathrm{T}(Ay_i - b)\right\|^2 + \tau \|x_i\|_1\,, \tag{3}$$

subject to $x_i \geq 0$, which is minimized by

$$x_i = \max\left(\frac{\frac{L^2}{2} y_i - LA^\mathrm{T}(Ay_i - b) - \tau}{\frac{L^2}{2}}, 0\right). \tag{4}$$

The results of the projection operations A and A^T are cached; in addition, Ay_i is computed from Ax_i and Ax_{i-1}, so that both the forward and backward projections are only executed once during each iteration.

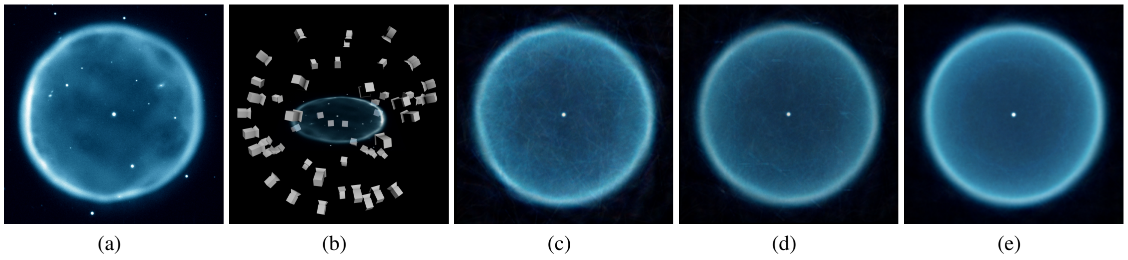

Fig. 6. The planetary nebula Abell 39 and reconstructions assuming spherical symmetry. The original image (a) was replicated at several virtual camera positions distributed randomly around the center of the object (schematic display in (b)) and subsequently reconstructed at a resolution of 512^3 voxels with $\tau = 0$ and $n_{\text{outer}} = 40$ for 64 (c), 128 (d), and 512 (e) virtual cameras in 3 539 sec, 6 950 sec, and 33 034 sec, respectively. *Original image: WIYN/NOAO/NSF.*

As a fundamental extension, we allow for hard constraints of the form $Bx = c$, with $BB^{\mathrm{T}} = \mathbb{I}$. Such constraints are useful if, for example, one view is known to be exact, but all others are subject to noise or inconsistencies, as for astronomical data where one projection is known precisely but all others are highly speculative. In that case, B would be chosen as a single projection M_k. The constraints are approximated by alternately projecting onto the subspaces of feasible solutions and of nonnegative solutions in an inner loop comprising n_{inner} iterations. Again, the number of iterations specifies the tradeoff between runtime and compliance with the constraints, providing a means to control the level of similarity. Alternatively, the largest acceptable difference between Bx and c can be provided as a termination criterion. This step is completely optional and does not entail any performance penalties when no constraints are specified.

For practical reconstruction problems, additional prior information is often given as an approximate a priori assumption about the distribution of intensity. For example, if an object is known to be compact, the presence of intensity farther from the center is increasingly unlikely. We incorporate such prior information in the reconstruction algorithm by formally replacing the scalar regularization parameter τ by a vector. Thus, a different regularization parameter can be specified for each voxel in x, where smaller values of τ represent a higher a priori probability of intensity in the corresponding voxel. This kind of spatially dependent regularization can lead to much more compact and realistic models with less background noise.

5 Forward and Backward Projection

For the actual reconstruction, we need to repeatedly project the images to the volume and vice versa using the sparse matrices M_k and M_k^T. Assuming the size of our input images is n^2 and the size of the volume is n^3, each M_k consists of n^5 entries with about n^3 nonzero entries. Since we need all matrices M_k in each iteration and the number of images is of the same order as n, we need to store a total number of about n^4 nonzero entries out of a total number of about n^6. The total storage requirement for $n = 1024$ would easily exceed a couple of terabytes just for the nonzero values. Thus, instead of calculating and storing M_k explicitly, we calculate $M_k v$ and $M_k^T b_k$ for each iteration.

Assuming that the scalar voxel values of the volume define a sampled piecewise trilinear function in space, each value of v has a footprint that is twice as large as the distance between two voxels along all three axes. Since the footprint is symmetrical along the axes, we can use a common formula to integrate one grid cell, i.e., the space between eight neighboring sampling points, at a time. Since we consider an emission-only model, the weighted sample values of the eight neighboring voxels are simply added up and we can solve the integration on a per-voxel basis.

To solve the integration analytically, we consider the influence of one voxel v_i on a single grid cell. Without loss of generality, we place the voxel at position $(1,1,1)$ in a local coordinate system that is rotated such that the grid cell in question coincides with the unit cube with coordinates in the $[0,1]$ range. The scalar value at any sample point s inside this unit cube is therefore $v_i s_x s_y s_z$. Since we need to integrate the scalar values along a ray intersecting the unit cube, we define e as the entry point, d as the direction, and l as the length of the intersecting ray segment. A sample point can thus be represented as $s = e + td$ with $0 \le t \le l$. The integrated emission can then be calculated as $v_i m$ with m defined as

$$\begin{aligned} m &= \int_0^l (e_x + td_x)\left(e_y + td_y\right)(e_z + td_z)\,dt \qquad (5)\\ &= e_x e_y e_z l + \frac{d_x e_y e_z + e_x d_y e_z + e_x e_y d_z}{2} l^2 + \\ &\quad \frac{e_x d_y d_z + d_x e_y d_z + d_x d_y e_z}{3} l^3 + \frac{d_x d_y d_z}{4} l^4 . \end{aligned}$$

Note that each of the entries of M_k is the sum of m over all affected grid cells for a given v_i.

To avoid scatter write and write collision in a parallel implementation, we can simply calculate one scalar output of the matrix multiplication in a single thread. The forward projection $M_k v$ uses the fast voxel traversal algorithm by Amanatides and Woo [1] to find all voxel values v that contribute to a single pixel of the image b_k as shown in Figure 4(a). As part of our contribution, we combine this algorithm with our analytically integrated kernel for trilinear interpolation to implement an efficient and accurate parallel GPU raycaster that traverses the entire volume along several viewing rays at once before writing the accumulated result to the output image.

Parallelizing over the output again, the backward projection $M_k^T b_k$ needs to find all pixels of the image b_k that contribute to a given voxel v. We therefore project the bounding box of the $2 \times 2 \times 2$ grid cells to the input image and cast a ray for each pixel through the small subvolume as shown in Figure 4(b). The same integration as above is used again, but this time we multiply m with the corresponding pixel in b_k. This can be seen as constructing the matrices M_k in column order instead of row order.

6 Distributed Architecture

After having discussed our optimization algorithm and the projection operators in the previous sections, we now contribute an implementation for a distributed multi-GPU cluster. In Section 2, we illustrated that memory requirements grow significantly for large datasets and clearly exceed the available resources on a single GPU, which motivates our approach to employ a distributed environment. Therefore, we seek to achieve two goals by adding more compute nodes to the system: high data scalability for large volumes by exploiting the combined graphics memory in a distributed cluster (including multiple GPUs in one physical node), and reasonable performance scalability.

In the following, we refer to a *compute node* as a process that has exclusive access to a dedicated GPU, i.e., the number of compute nodes equals the total number of available GPUs in the cluster domain. We start our approach by decomposing the volume data V into sub-volumes V_i of equal size and distribute them evenly across all compute nodes. The sub-volumes V_i are padded with an additional layer

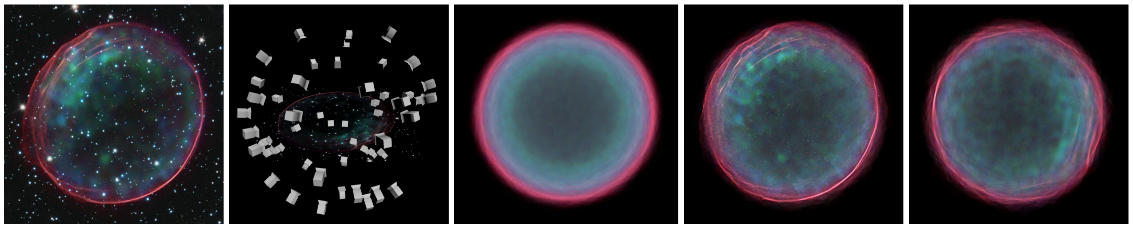

Fig. 7. The supernova remnant 0509-67.5 and reconstructions assuming spherical symmetry. After manually removing the background stars, the original image (a) was replicated at 128 virtual camera positions distributed randomly around the center of the object (schematic display in (b)). Without constraining the projection from the front, details are averaged out, leading to an overly symmetric model (c). Constraining the projection from the front to be similar to the original image preserves asymmetric features and details and accurately reproduces the original view (d), whereas a different view (e) exhibits new aspects but retains similarity to the coarse structure of the original view. The volume was reconstructed at a resolution of 512^3 voxels with $\tau = r \cdot 10^3$, $n_{\text{outer}} = 40$, and $n_{\text{inner}} = 2$ in 8 592 sec, where r is the Euclidean distance from each voxel to the center of symmetry. *Original image: NASA, ESA, CXC, SAO, the Hubble Heritage Team (STScI/AURA), and J. Hughes (Rutgers University).*

of voxels at the boundaries to ensure seamless transitions between adjacent bricks of data. The initial image data for the reconstruction is replicated on each compute node in a startup phase.

The computational steps of our algorithm in Figure 3 consist of basic vector operations as well as the operators A and A^{T}, and, when constraints are specified, B and B^{T}. The componentwise vector operations are executed independently in parallel on each GPU as they do not require any communication. However, the distribution of the projection steps requires more attention to detail. An overview of both projections is illustrated in Figure 5. Each compute node i performs the forward projection from Section 5 of its sub-volume V_i on the GPU for all view points $k = 0, \ldots, n_{\text{views}} - 1$ and obtains a set of partial images P_i^k as shown in Figure 5(a). In the next step, the partial images of each view point need to be composited. From sort-last volume rendering, it is well-known that compositing can be a severe bottleneck, even for a single image. In our case, the number of images can be quite large, up to a few hundreds, which further pushes computational demands and communication overhead.

To accelerate compositing, we suggest a two-step approach. First, the computational workload is distributed evenly across all compute nodes. Assuming n_{views} view points and n_{nodes} compute nodes, the total compositing workload C is:

$$C = \sum_{k=0}^{n_{\text{views}}-1} \sum_{i=0}^{n_{\text{nodes}}-1} P_i^k \tag{6}$$

The outer sum can be computed in parallel by using a round-robin scheme, i.e., each node i is assigned a partial sum. However, not all partial images of one view point are available on each node. Therefore, the missing images need to be transferred over the network according to the decomposition of the sum. Afterward, the inner sum in Equation 6 is computed on each GPU as shown in Figure 5(b).

Until now, there is a composited image for each view point, but the result is spread all over the nodes. However, for subsequent operations with the entire image data, e.g., the backward projection, it is necessary to provide the result on each node. Therefore, the second step of our compositing algorithm is a final scattering of the image data over the network as shown in Figure 5(c). In contrast to the forward projection, the backward projection does not require a ray traversal of the entire volume but only of a small region of $2 \times 2 \times 2$ grid cells; hence it can be processed independently on each node, once the image data is available.

With the described partitioning scheme, our algorithm scales with growing data size by adding more nodes to the domain since the forward projection is similar to sort-last rendering. However, since we render up to a few hundred images, compositing becomes a bottleneck. We address this issue by distributing the workload among the compute nodes. In fact, this multi-compositing step is similar to a sort-first approach by considering the images from all view points as the tiles of a very large virtual image. Each node computes a part of this virtual image in parallel by additive compositing. For the backward projection, the entire virtual image is required on each node, which can be considered as the final gathering step in a sort-first approach with multiple view points.

7 Results

To evaluate the visual quality of the results and the performance of the proposed algorithm, we present reconstructions of approximately spherically and axially symmetric nebulae. Direct volume rendering is used to visualize the resulting volumetric data. Fly-by animations of the reconstructed objects can be found in the supplementary material. The parallel algorithm was executed on a GPU cluster consisting of 32 physical nodes, each with 2 Intel Xeon X5620 Quad Core CPUs, 2 Nvidia GeForce GTX480 GPUs, and 24 GB RAM. The physical nodes are interconnected over an InfiniBand network with a bandwidth of 20 GBit/s. The parallel implementation employs C++ for the host code, CUDA for the GPU code, and mvapich2 for the communication via MPI. An MPI process is deployed for each GPU in the cluster domain to support flexible execution configurations.

As a first example, we consider the planetary nebula Abell 39, Figure 6(a). Its geometry resembles a hollow sphere. For its reconstruction, virtual cameras were placed at random locations around the center, Figure 6(b). By associating the original image with all of these virtual views, the assumption of spherical symmetry is implicitly defined. The corresponding reconstruction reproduces the supposed geometry of the object with increasing accuracy as the number of projections increases, Figures 6(c)–(e). Since the object is of almost perfect spherical symmetry, the projections are largely consistent and no regularization is needed; so we set $\tau = 0$.

The supernova remnant 0509-67.5 is a nebula with only approximate spherical symmetry, Figure 7(a). In the false-color image, visible-light observations from Hubble Space Telescope (pink and surrounding star field) are combined with X-ray data from Chandra X-ray Observatory (blue and green). This example illustrates how the proposed algorithm handles arbitrary projection geometries, massively inconsistent projections, equality constraints, and spatial regularization. We again use a setup implementing spherical symmetry, see Figure 7(b). Since the symmetry is only approximate, the projections are inconsistent, and without further precautions details would be averaged out, Figure 7(c). To preserve the familiar appearance of the object from the initial perspective, we add an equality constraint. To resolve the ambiguity introduced by the competing projections, we make use of location-dependent regularization by choosing τ as a function of position. On the one hand, regularization reduces the amount of voxels with nonzero intensity, thereby suppressing typical artifacts (see also Figure 10). On the other hand, with τ increasing radially from the center, compact objects are favored, similar to the implicit regularization mechanism used for reconstruction of reflection nebulae [18]. The

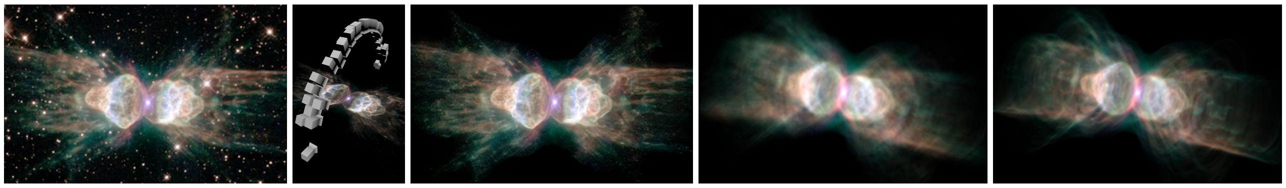
(a) (b) (c) (d) (e)

Fig. 8. The Ant Nebula (Mz 3), and reconstructions assuming axial symmetry. After manually removing the background stars, the original image (a) was replicated at 128 virtual camera positions distributed randomly around the symmetry axis of the object (schematic display in (b)). To obtain a more natural and less symmetric impression, the symmetry axis was jittered by $\pm 4^\circ$ for each camera. Constraining the projection from the front to be similar to the original image, the volume was reconstructed at a resolution of 512^3 voxels with $\tau = r \cdot 10^3$, $n_{\text{outer}} = 40$, and $n_{\text{inner}} = 5$ in 9 633 sec, where r is the Euclidean distance from each voxel to the symmetry axis. The resulting view from the front (c) closely resembles the original image. The oblique view (d) exhibits less detail but an overall realistic shape. In contrast, the same view of a model reconstructed from cameras distributed uniformly around the axis without jittering (e) looks less realistic (especially when animated) and suffers from directional artifacts. *Original image: NASA, ESA, and The Hubble Heritage Team (STScI/AURA).*

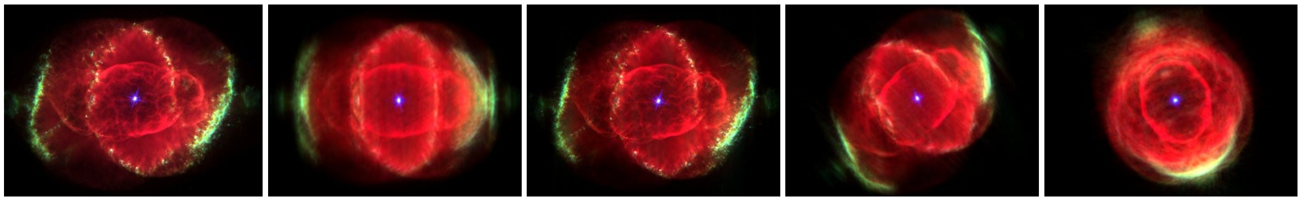
(a) (b) (c) (d) (e)

Fig. 9. The Cat's Eye Nebula (NGC 6543) and reconstructions assuming axial symmetry. The original image (a) was replicated at 128 virtual camera positions distributed randomly around the symmetry axis of the object as in Figure 8(b), again with jittering about $\pm 4^\circ$ at a resolution of 512^3 voxels with $\tau = r \cdot 10^3$. Without constraining the projection from the front, the result is overly symmetric and looks unrealistic (b). Employing our constrained approach with $n_{\text{inner}} = 5$ resembles the original image more convincingly as it introduces asymmetric features (c). Rotating the constrained model toward the axis of symmetry still shows asymmetric features (d). As the vantage point approaches the symmetry axis, the apparent shape of the Cat's Eye nebula changes more toward a ring nebula (e), in contradiction with the canonical interpretation which is backed by additional data. *Original image: J.P. Harrington and K.J. Borkowski (University of Maryland), and NASA.*

result is a consistent and plausible volumetric visualization that is approximately symmetric but retains its resemblance to the original, see Figure 7(d), as well as a high amount of realistic, fine-grained detail for other vantage points, see Figure 7(e).

The Butterfly Nebula, or M2-9, is an example of a bipolar planetary nebula whose structure is more easily described by an approximate axisymmetry, see Figure 1(a). The axisymmetry can be modeled by distributing the virtual cameras randomly around the axis of symmetry. Only projections from the front are used; projections from the back would be equivalent except for mirroring of the image. Again, the projection from the front is constrained to be similar to the observed image, and the regularization weight τ increases with distance from the symmetry axis. Even though the assumed symmetry is only approximate, most details are clearly visible in the reconstructed volume, see Figure 1(b). When the point of view approaches the axis, the two-shell structure of the nebula becomes apparent although some detail is inevitably averaged out, see Figures 1(c) and 1(d).

The Ant Nebula, or Mz 3, is another example of a bipolar nebula, albeit with much more fine structure and less apparent symmetry, see Figure 8(a). The cameras are again arranged around the symmetry axis; to increase the amount of perceived three-dimensionality, the axis is randomly inclined for each camera so that moving about the axis produces more visual variation, see Figure 8(b), We again use spatial regularization and an equality constraint to preserve the original appearance, see Figure 8(c). When seen from a novel viewpoint, see Figure 8(d), our visualization shows much more detail and less visible artifacts than previous approaches. The result is more realistic than a reconstruction from equidistant cameras, see Figure 8(e).

To demonstrate the importance of equality constraints, we consider the Cat's Eye Nebula, or NGC 6543, see Figure 9(a). It is a rather complex nebula whose shape is believed to consist mainly of an elongated central bubble and two larger spherical lobes. Due to its asymmetry, a simple axisymmetry assumption produces overly symmetric, unrealistic results, see Figure 9(b). Using an equality constraint for the original projection reproduces the nebula much more accurately, see Figure 9(c). Novel views, however, reveal that the alleged bispherical geometry is only imperfectly reconstructed; the reconstructed geometry instead resembles a single larger shell, see Figures 9(d) and (e).

To study the effects of different amounts of regularization, we reconstruct the planetary nebula NGC 6826, see Figure 10(a), with different parameters assuming axial symmetry. Without regularization, artifacts arise in the outer regions of the volume, see Figure 10(b). Moderate regularization entirely removes these artifacts, see Figure 10(c), and produces a faithful visualization, see Figure 10(d), that looks plausible also from novel viewpoints, see Figure 10(e). Although the range of suitable values of τ comprises several orders of magnitude, excessively large values can lead to darkening of the outer parts of the object, see Figure 10(f). In practice, the same value of τ is appropriate for a wide range of objects, and the effects of too small or too large a value are easily recognized by comparison with the original image.

Figure 11 shows a failure case of our algorithm. The outer regions of the Butterfly nebula contain both absorption and scattering, neither of which are compatible with our basic assumptions. These regions are therefore not interpreted as emissive regions attenuated by an absorbing layer, but as empty space. While the original view can be reproduced, see Figure 11(b), oblique views exhibit missing parts, and their appearence differs significantly from the expected structure, see Figures 11(c) and (d).

We verify the accuracy of our algorithm and its applicability to general tomographic reconstruction problems by reconstructing a known test dataset, see Figure 12(a), from a number of CT-like projections. We observe that the reconstruction error declines approximately exponentially with the number of steps, see Figure 12(c), and that the rel-

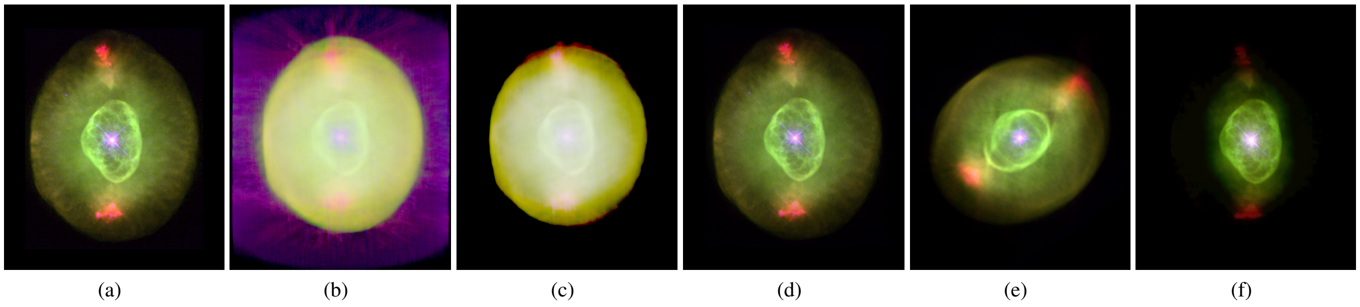

Fig. 10. Analysis of regularization parameter τ with reconstructions of planetary nebula NGC 6826. The original image (a) was replicated at 128 virtual camera positions distributed randomly around the symmetry axis of the object, again with jittering about $\pm 4^\circ$. Without regularization, by setting $\tau = 0$, noise and streaking are evident, here displayed with logarithmically scaled intensity to make them clearly visible (b). Employing our regularization approach with $\tau = r \cdot 10^3$, where r is the Euclidean distance from each voxel to the symmetry axis, reduces noise and streak artifacts significantly at the same intensity scaling (c). With linearly scaled intensity, the reconstruction closely reproduces the original image (d) and also remains plausible when seen from a different vantage point (e). In contrast, if the regularization factor is chosen much too large (here $\tau = r \cdot 5 \cdot 10^4$), the outer parts of the nebula become suppressed (f). *Original image: Bruce Balick (University of Washington), Jason Alexander (University of Washington), Arsen Hajian (U.S. Naval Observatory), Yervant Terzian (Cornell University), Mario Perinotto (University of Florence, Italy), Patrizio Patriarchi (Arcetri Observatory, Italy), and NASA*.

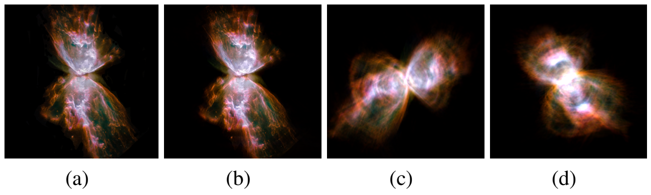

Fig. 11. The Butterfly Nebula (NGC 6302) and reconstructions assuming axial symmetry. The original image (a) was replicated at 128 virtual camera positions distributed randomly around the symmetry axis of the object as in Figure 8(b), again with jittering about $\pm 4^\circ$ at a resolution of 512^3 voxels with $\tau = r \cdot 10^3$. This is a typical failure case for our algorithm as the large amounts of dust in the nebula violate the assumption of pure emission. While the rendering from the original vantage point is able to reproduce the input image accurately (b), oblique views do not reproduce the expected absorption at the tips of the two main lobes (c)–(d). *Original image: NASA, ESA, and the Hubble SM4 ERO Team*.

ative squared error $e = \|x - x'\|^2 / \|x\|^2$ after 100 steps is of the order of 10^{-3}, where x is the original volume and x' is our reconstruction. The reconstructed image shows no visible artifacts except for a slight smoothing, see Figure 12(b). This accuracy seems to be sufficient to allow for applications of the algorithm for our problem as well as in other fields.

To quantitatively validate our method in its original application domain, we apply it to a single projection, see Figure 13(a), of a protoplanetary nebula model built manually by an astronomer, see Figure 13(b). The numerical comparison of the reconstructed volume to the original dataset shows a relative error of about 4.6 %. The value is expected to be higher for nebulae with a less pronounced symmetry. An oblique view of the reconstructed model, see Figure 13(c), shows some weak streaking artifacts that would likely be suppressed by jittering of the axis, albeit at the cost of a larger numerical error.

In Figure 14, we plot the computational time of one iteration step for different data sizes, ranging from 64^3 to 1024^3 voxels, and for varying numbers of views as a function of the number of nodes. The plots allow for two observations to be made. For large datasets, a larger number of nodes is necessary to handle the amount of data without expensive sequential processing and data transfer. For very small datasets, however, a high number of nodes is counterproductive because the padding of the sub-volumes reaches a significant portion of the overall data and the time used for network communication and synchronization becomes dominant.

Achieving optimal speedups in parallel algebraic reconstructions

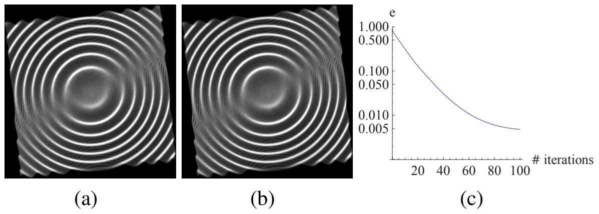

Fig. 12. The Marschner-Lobb test dataset [22] (a) and our reconstruction (b) at 128^3 voxels resolution from 128 virtual cameras distributed evenly around the up axis. The relative squared error $e = \|x - x'\|^2 / \|x\|^2$ is computed from the original volume x and our reconstruction x', and is plotted logarithmically over the number of steps (c). This experiment demonstrates the accuracy of our algorithm in a CT-like setting.

is an inherently difficult problem due to strong data dependencies. Melvin et al. [24] discuss this issue in detail for classic CT reconstruction in medicine. The authors' solution to reducing the impact of communication overhead is a highly-specialized shared-memory hardware architecture. With this setup, a speedup of about 21x could be achieved with 32 CPU-based nodes. In our approach, we employ commodity hardware and we build upon GPU-accelerated nodes with a fully distributed memory architecture. Although the high performance–to–cost ratio of multiple GPUs is attractive compared to such a dedicated HPC cluster, latency induced by network-based communication combined with the SIMD parallelization of the GPUs has a negative impact on scalability. This observation is accompanied by the results from Jang et al. [13], who iteratively reconstruct a least-squares approximation on an Nvidia Tesla S870 with 4 GPUs. Although the authors report high speedups compared to a CPU-based implementation, the performance gain of using 4 GPUs as compared to 1 GPU reaches only a factor of about 2.0x for the forward projection and 2.7x for the backward projection, including overhead due to data transfer and synchronization respectively. According to Figures 14(a)–(c), we achieve comparable speedups of 2.2x–3.2x with 4 GPUs, but in a distributed environment.

We quantify the overhead of communication for a data size of 256^3 voxels and 256 viewpoints in Figure 15 with a detailed breakdown of the fractional times within one iteration step. With an increasing number of nodes, "Compositing Gather" becomes predominant, which is the first transfer step in the distributed compositing algorithm, Figure 5(b). For more than 32 nodes, communication becomes dominant and performance scalability reaches saturation for this particular parameter setup, see Figure 14.

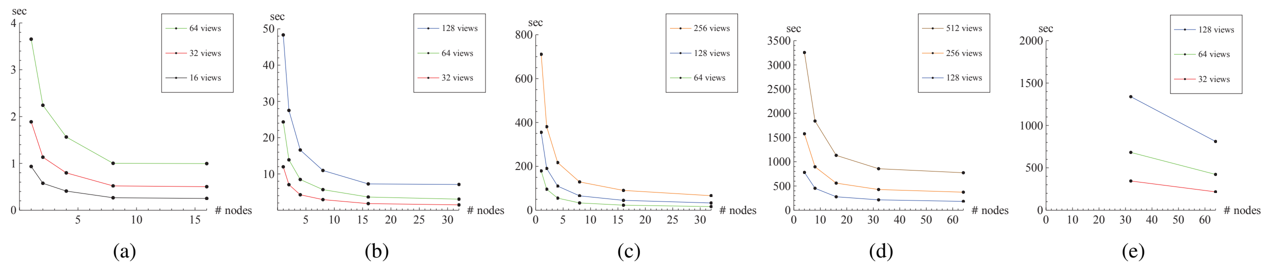

Fig. 14. Time required for one iteration step at volume sizes of 64^3 (a), 128^3 (b), 256^3 (c), 512^3 (d), and 1024^3 (e) voxels with varying number of virtual views. For 512^3 voxels, a minimum of 4 nodes is required to handle the data size. For 1024^3 voxels, 32 or more nodes are required, and the number of views is bounded to 128 due to limited memory.

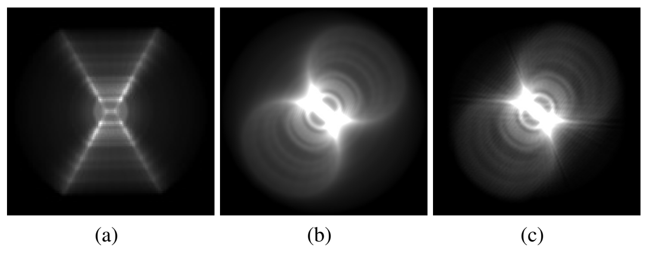

Fig. 13. A simplified (emission-only) model of the Red Rectangle nebula was reconstructed from a single projection (a) at 128^3 voxels resolution from 128 virtual cameras distributed evenly around the up axis. Since the nebula fulfills the symmetry assumption well, the reconstruction (c) is visually quite similar to the original (b), and the relative error, cf. Figure 12, is only about 4.6 %. *Model courtesy of Koning et al. [15].*

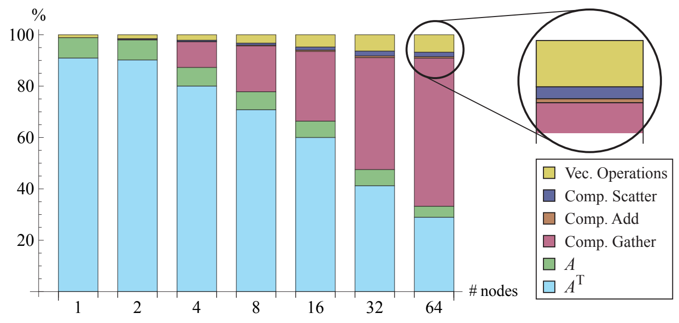

Fig. 15. Detailed breakdown of one iteration step for a volume with a resolution of 256^3 voxels and 256 virtual views. The projections A and A^T and the vector operations are predominant until the number of nodes reaches 32, when communication cost exceeds computation time.

8 Conclusion and Future Work

We have shown that our tomographic algorithm is capable of reconstructing volumes of resolutions up to 1024^3 voxels from up to 512 projections in a fully automatic way. When presented with strongly inconsistent, contrived projections in the context of astronomical nebula reconstruction from single images, a regularization scheme preserves the plausibility of the result.

Astronomical nebulae of roughly spherical shape were shown to be satisfyingly reconstructed by the algorithm, retaining a high amount of detail and present irregularities in the geometry. Axisymmetric objects, on the other hand, lose some detail when the camera approaches the symmetry axis. Here, the generation of synthetic detail may provide a remedy in the future. However, our additional constraints not only help reproduce the original view convincingly, but also reconstruct crucial asymmetric features that convey a basic impression of irregularity even when the viewpoint is close to the axis.

Since our model only reconstructs emission, nebulae that contain a significant amount of scattering or absorption are reconstructed poorly. Reconstructions including simultaneous emission and absorption require nonlinear optimization and are in general more computationally intensive. The *attenuated ray transform* [29] may provide a starting point for a model comprising emission and absorption but no scattering. One notes that the change in the intensity $I(x)$ along the viewing ray in the presence of absorption $a(x)$ and emission $e(x)$ is described by $\mathrm{d}I(x)/\mathrm{d}x = -a(x)I(x) + e(x)$, which is solved by $I(x) = I(0)\exp\left(-\int_0^x a(t)\mathrm{d}t\right) + \int_0^x e(t)\exp\left(-\int_t^x a(s)\mathrm{d}s\right)\mathrm{d}t$. The inverse problem could in principle be solved using ℓ_1 minimization algorithms [4]. Recent grid-free methods [11] may also provide a method to solve this problem efficiently. In contrast, if scattering is taken into account, every voxel potentially influences the intensity of every image pixel, and the inherent sparsity of the projection operator M is lost. High-resolution reconstruction of scattering nebulae may be possible using multi-resolution methods that are already used for rendering [18]. In cases where the scattering effects can be approximated by a convolution in the image plane, the problem could probably be solved by a modified version of our algorithm, but convergence is likely to be considerably slower.

The quality of the reconstruction is naturally limited by the fact that the algorithm has no knowledge about the physical processes underlying the objects being reconstructed. Since our algorithm provides a mechanism for specifying additional constraints in a generic way, it would be possible to restrict the search space to solutions compatible with a physical model. Additionally, an interactive volumetric reconstruction tool could let the user guide the automatic reconstruction by specifying the position of substructures in space or by manipulating individual views. Obviously, an interactive editor would require further acceleration of the reconstruction algorithm and live display of intermediate results. Higher performance could be achieved by means of a more sophisticated compositing scheme and additional optimizations in the projection kernels, e.g. through the use of more efficient memory caching. In addition, a multi-resolution workflow would allow us to reconstruct a low-resolution model quickly in interactive mode while high-resolution detail would be synthesized in a separate offline step. In the long run, we seek to utilize the presented algorithm as the foundation for a versatile and intuitive interactive, constraint-based volumetric modeling framework that may also find applications outside astronomical visualization as well.

Acknowledgments

The authors would like to thank the reviewers for their insightful comments and constructive criticism. This work was partially funded by Deutsche Forschungsgemeinschaft (DFG) as part of DFG projects MA 2555/7–1 and WE 2836/2–1 ("Astrographik") as well as LO 1436/3–1 and PF 709/1–1 ("Sparse Exact and Approximate Recovery").

References

[1] J. Amanatides and A. Woo. A fast voxel traversal algorithm for ray tracing. *Proceedings of Eurographics*, pages 3–10, 1987.

[2] R. Baraniuk. Compressive sensing. *IEEE Signal Processing Magazine*, 24(4):118–121, 2007.

[3] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[4] K. Bredies, T. Bonesky, D. A. Lorenz, , and P. Maass. A generalized conditional gradient method for non-linear operator equations with sparsity constraints. *Inverse Problems*, 23:2041–2058, 2007.

[5] E. J. Candès, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59:1207–1223, 2006.

[6] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52:1289–1306, 2006.

[7] S. Eilemann and R. Pajarola. Direct send compositing for parallel sort-last rendering. In *Proceedings of the Eurographics Symposium on Parallel Graphics and Visualization*, pages 29–36, 2007.

[8] T. García-Díaz, J. A. López, W. Steffen, M. G. Richer, and H. Riesgo. A Cat's Eye view of the Eskimo from Saturn. In *Proceedings of the IAU Symposium 283 "Planetary Nebulae: An Eye to the Future"*, 2011.

[9] G. Golub and C. Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 1996.

[10] R. Gordon, R. Bender, and G. T. Herman. Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography. *Journal of Theoretical Biology*, 29(3):471–481, 1970.

[11] J. Gregson, M. Krimerman, M. B. Hullin, and W. Heidrich. Stochastic tomography and its applications in 3D imaging of mixing fluids. *ACM Transactions on Graphics*, 31(4):52:1–52:10, 2012.

[12] I. Ihrke and M. Magnor. Image-based tomographic reconstruction of flames. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 365–373, 2004.

[13] B. Jang, D. Kaeli, S. Do, and H. Pien. Multi GPU implementation of iterative tomographic reconstruction algorithms. In *Proceedings of the IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 185–188, 2009.

[14] X. Jia, Y. Lou, R. Li, W. Song, and S. Jiang. GPU-based fast cone beam CT reconstruction from undersampled and noisy projection data via total variation. *Medical Physics*, 37:1757–1760, 2010.

[15] N. Koning, S. Kwok, and W. Steffen. Morphology of the Red Rectangle proto-planetary nebula. In *Proceedings of the IAU Symposium 283 "Planetary Nebulae: An Eye to the Future"*, 2011.

[16] D. A. Leahy. Deprojection of emission in axially symmetric transparent systems. *Astronomy and Astrophysics*, 247:584–589, July 1991.

[17] K.-L. Ma, J. S. Painter, C. D. Hansen, and M. F. Krogh. Parallel volume rendering using binary-swap compositing. *IEEE Computer Graphics and Applications*, 14:59–68, July 1994.

[18] M. Magnor, K. Hildebrand, A. Lintu, and A. Hanson. Reflection nebula visualization. *Proceedings of IEEE Visualization*, pages 255–262, 2005.

[19] M. Magnor, G. Kindlmann, C. Hansen, and N. Duric. Constrained inverse volume rendering for planetary nebulae. In *Proceedings of IEEE Visualization*, pages 83–90, 2004.

[20] M. Magnor, G. Kindlmann, C. Hansen, and N. Duric. Reconstruction and visualization of planetary nebulae. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):485–496, Sept. 2005.

[21] S. Marchesin, C. Mongenet, and J.-M. Dischler. Multi-GPU sort-last volume visualization. In *Proceedings of the Eurographics Symposium on Parallel Graphics and Visualization*, pages 1–8, 2008.

[22] S. R. Marschner and R. J. Lobb. An evaluation of reconstruction filters for volume rendering. In *Proceedings of IEEE Visualization*, pages 100–107, 1994.

[23] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[24] C. Melvin, M. Xu, and P. Thulasiraman. HPC for iterative image reconstruction in CT. In *Proceedings of the 2008 C^3S^2E conference*, pages 61–68, 2008.

[25] S. Molnar, M. Cox, D. Ellsworth, and H. Fuchs. A sorting classification of parallel rendering. *IEEE Computer Graphics and Applications*, 14(4):23–32, July 1994.

[26] B. Moloney, M. Ament, D. Weiskopf, and T. Möller. Sort-first parallel volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 17(8):1164–1177, 2011.

[27] C. Müller, M. Strengert, and T. Ertl. Optimized volume raycasting for graphics-hardware-based cluster systems. In *Proceedings of the Eurographics Symposium on Parallel Graphics and Visualization*, pages 59–66, 2006.

[28] D. Nadeau, J. Genetti, S. Napear, B. Pailthorpe, C. Emmart, E. Wesselak, and D. Davidson. Visualizing stars and emission nebulas. *Computer Graphics Forum*, 20(1):27–33, 2001.

[29] F. Natterer and F. Wübbeling. *The attenuated ray transform*, chapter 2.4.1. Society for Industrial Mathematics, 2001.

[30] X. Pan, E. Sidky, and M. Vannier. Why do commercial CT scanners still employ traditional, filtered back-projection for image reconstruction? *Inverse Problems*, 25(12):123009, 2009.

[31] T. Peterka, D. Goodell, R. Ross, H.-W. Shen, and R. Thakur. A configurable algorithm for parallel image-compositing applications. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, SC '09, pages 4:1–4:10, 2009.

[32] F. Sabbadin, M. Turatto, R. Ragazzoni, E. Cappellaro, and S. Benetti. The structure of planetary nebulae: theory vs. practice. *Astronomy and Astrophysics*, 451(3):937–949, 2006.

[33] E. Sidky and X. Pan. Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization. *Physics in Medicine and Biology*, 53:4777, 2008.

[34] W. Steffen, N. Koning, S. Wenger, C. Morisset, and M. Magnor. Shape: A 3D modeling tool for astrophysics. *IEEE Transactions on Visualization and Computer Graphics*, 17(4):454–465, Apr. 2011.

[35] J. Tang, B. E. Nett, and G.-H. Chen. Performance comparison between total variation (TV)-based compressed sensing and statistical iterative reconstruction algorithms. *Physics in Medicine and Biology*, 54(19):5781–5804, 2009.

[36] S. Wenger, J. Aja Fernández, C. Morisset, and M. Magnor. Algebraic 3D reconstruction of planetary nebulae. *Journal of WSCG*, 17(1):33–40, Feb. 2009.

[37] S. Xiao, Y. Bresler, and D. Munson Jr. Fast Feldkamp algorithm for cone-beam computer tomography. In *Proceedings of the International Conference on Image Processing*, volume 2, pages II–819, 2003.

[38] W. Zheng and C. O'Dell. A three-dimensional model of the Orion nebula. *Astrophysical Journal*, 438(2):784–793, Jan. 1995.