UNIVERSITY NAME

DOCTORAL THESIS

Thesis Title

Author:
John SMITH

Supervisor:
Dr. James SMITH

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy

in the

Research Group Name
Department or School Name

June 26, 2018

Declaration of Authorship

I, John SMITH, declare that this thesis titled, "Thesis Title" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

UNIVERSITY NAME

Abstract

Faculty Name

Department or School Name

Doctor of Philosophy

Thesis Title

by John SMITH

-ct scans / MRi

– rendering technique

– interaction

– occlusion management

– VR and mixed reality

– Use cases

V…

Acknowledgements

The acknowledgments and the people to thank go here, don't forget to include your project advisor...

Contents

List of Figures

List of Tables

List of Abbreviations

LAH	**L**ist **A**bbreviations **H**ere
WSF	**W**hat (it) **S**tands **F**or

Physical Constants

Speed of Light $c_0 = 2.997\,924\,58 \times 10^8\,\mathrm{m\,s^{-1}}$ (exact)

List of Symbols

a	distance	m
P	power	W ($\mathrm{J\,s^{-1}}$)
ω	angular frequency	rad

For/Dedicated to/To my…

Chapter 1

Introduction

Volumetric data sets are present in many fields such as medical imaging, physics, natural science, security, engineering etc. They are composed of voxels which represents the shape of these data. In fact, a volume is a scalar function of three spatial variables $(x, y, z) \in \mathbb{R}^3$. Voxel volumes can be produced by 3 main ways. The first one is to directly acquire from the real world thanks to some specific devices. As an example, X-ray scanners allow to collect data from baggage in airports. The second method is to generate the voxels through the volume by using mathematical models. For instance, a fluid flow in a basin can be represented in a volume of voxels thanks to the adequate mathematical model. The third way to produce volumetric data sets is to rasterize vector data model using algorithms. This can be carried out for many purposes such as retrieving new insights thanks to different visualization techniques.

To visualize these type of data sets, different rendering algorithms can be used. There are two major approaches to volume rendering. The first approach is to use ray casting based algorithms. They directly come from the rendering equation. They consists in shooting rays for each pixel of the final rasterized 2D image, sampling along the part of the ray located inside the volume, shading the sampling points, and compositing all the sampling points. The second approach to render volumetric data sets is to use plane compositing. It consists in accumulating information over the whole view plane for each plane of voxels in the data set. When each plane is processed, a pixel of the final rasterized 2D image is updated. This technique is texture-based and uses slices of the 3D Volume. Theses slices can be either aligned with the data set or with the viewing plane.

However, interacting with such data sets is not trivial. In fact, 3D volume visualization face many challenges such as the occlusion management and the computational time. In volume rendering, occlusion management is a challenge. As such, in 3d representations of volumes, some areas or objects (subsets) can be partially or fully hidden by others because of their locations. Transfer functions are used to match the volumetric data to colors in a meaningful way. Therefore, they are a good way to reduce occlusion and make visible interesting features. However, it is still difficult to create a good transfer function especially when the data are heterogeneous. In fact, designing a good transfer function depends heavily on the type of dataset and on the user's purpose. For instance, in the field of baggage inspection,

the variation of densities prevents to create a unique transfer function for each baggage. In contrast, it is easier to design a good transfer function for a system dedicated to visualizing the same type of datasets (brain CT scans, bone tissues, etc.).

Chapter 2

State of the art

2.1 Volume rendering algorithms

The fundamental volume visualization algorithms described below fall into two main categories:

- Direct volume rendering (DVR) algorithms: they include approaches such as ray-casting, texture mapping, splatting, and V-buffer rendering. Splatting and V-buffer are also called projection methods. These these methods are characterized by mapping elements directly into screen space without using geometric primitives as an intermediate representation.
- Surface-fitting (SF) algorithms : they are also called isosurfacing or feature-extraction. These algorithms consist in fitting surface primitives such as polygons or patches to constant-value contour surfaces in volumetric datasets. The surface primitives used are most of the time planar.

DVR methods are especially appropriate for creating images from datasets containing amorphous features like clouds, fluids, and gases. One disadvantage of using DVR methods is that the entire dataset must be traversed each time an image is rendered. A low resolution pass or random sampling of the data is sometimes used to create low-quality images quickly for parameter checking. The process of successively increasing the resolution and quality of a DVR image over time is called progressive refinement.

The user begins by choosing a threshold value and then geometric primitives are automatically fit to the high-contrast contours in the volume that match the threshold. Cells whose comer-values are all above the chosen threshold (cell is inside) or all below the threshold (cell is outside) are discarded and have no effect on the final image. Showing just the cells falling on the threshold is sometimes useful, but can be a problem. Another consideration is the huge number of surface primitives generated for large volumetric datasets.

Many steps in the volume visualization process are common to volume visualization algorithms. Most of the fundamental volume visualization algorithms include

only a subset of the steps listed here. The initial step in every procedure is data acquisition. The next common step is to pat the data slices into a form that can be worked with and then to process each slice so that it covers a good distribution of values, is high in contrast, and is free of noise and out-of-range values. Of course, the same set of processes must be applied to every slice. Next, the dataset is reconstructed so that the ratio of the dimensions is proportional to the ratio of the dimensions of the measured substance or substance being simulated. This may involve interpolating between values in adjacent slices to construct new slices, replicating existing slices, interpolating to estimate missing values, or scan-converting an irregular grid or non-orthogonal grid onto a Cartesian grid by interpolation. At this point three-dimensional enhancement may be applied, such as a slight blurring of the data values. Next, a data-classification or thresholding is performed. This step will be discussed in detail in the section on data classification. After data-classification, a mapping operation is applied to map the elements into geometric or display primitives. This is the stage that varies the most between algorithms, as will be shown in the section on volume visualization algorithms. At this point the primitives can be stored, manipulated, intermixed with externally defined primitives, shaded, transformed to screen space, and displayed. The shading and transformation steps can be reordered and shading can be done in one of eleven ways as explained in

2.1.1 Direct Volume Rendering

Texture mapping

2D Texture mapping 2D Texture mapping consists in representing the whole volume as a set of slices parallel to coordinate planes. To create images of the slices, one can use the standard texture mapping capabilities provided by graphics libraries like OpenGL. 2.1 shows how pixel colors and opacities are calculated when rendering a texture-mapped polygon. In texture mapping, texture coordinates (s, t) are stored along with each vertex. These coordinates are interpolated across the polygon during scan conversion, and are used as coordinates for color look-up inside a texture image 2.2. Color and opacity values from the texture image are reconstructed using bi-linear interpolation.

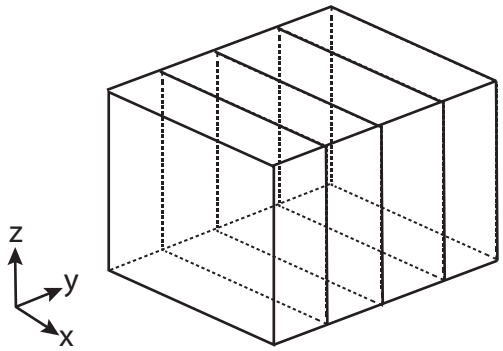


FIGURE 2.1: 2D slices of the volume.

On modern graphics workstations, the complete process of interpolating texture coordinates and reconstructing texture values is incorporated into the scan-conversion process and done completely in hardware. Because texture mapping is such a common application, these routines are well-optimized and highly efficient. To simulate ray casting using texture mapping, we have to slice the volumetric data set into parallel slices, and the use these slices as texture images to texture-map the projections of the sliced onto the image plane. To composite the slices, we can use the blending operations provided by the OpenGL graphics library. As it happens, one of the standard operators provided is exactly the compositing operator used in ray casting. Since compositing is also fully hardware-assisted, it is also very fast.

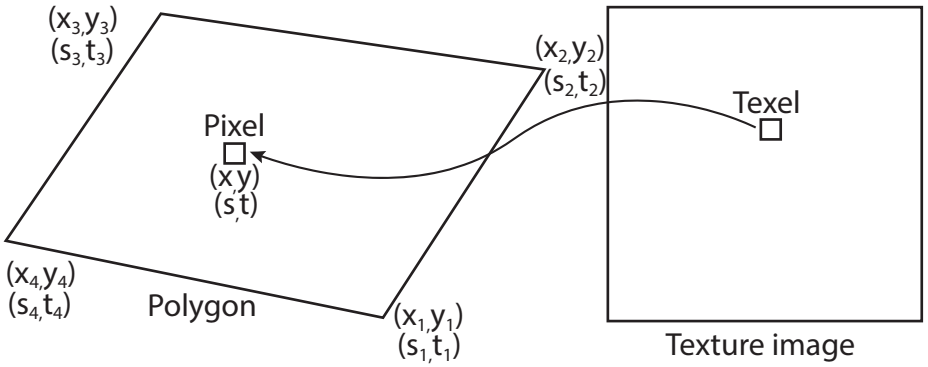


FIGURE 2.2: Calculating color and opacity of a pixel inside a texture-mapped polygon.

There are two ways to deal with varying viewpoints.

The first one is to create slice images for the given data set in a pre-processing step and uses different projection matrices to create arbitrary-viewpoint images of these slices. This is fast, since the (expensive) step of slice creation has to be done only once. On the other hand, image quality suffers because the views of all slices will be distorted due to the projection onto the image plane. In the extreme case of the view direction being parallel to the slices, there will be no image at all.

The second way is to create slices which are always orthogonal to the viewing direction. This results in optimal image quality, but it means that the slice stack has to be re-generated for each new image. Furthermore, arbitrary ("oblique") slices through a cuboid are generally not square or rectangular, and can be anything from triangles to hexagons.

3D Texture mapping To allow interactive generation of view-orthogonal slices, a special hardware technique has been developed. This is a generalization of texture-mapping to three-dimensional textures, appropriately called "3D texturing." As seen in Figure 2.2, 2D texture-mapping interpolates two additional coordinates (s, t) across a polygon's interior. In 3D texture-mapping, three additional coordinates (s, t, r) are interpolated. To determine a pixel's color and opacity, these three coordinates are used as indices into a three-dimensional image, the 3D texture, see Figure 2.3. To reconstruct texture values, tri-linear interpolation is used.

Algorithm 1 Volume Rendering using 2D Texture Mapping Algorithm

Require: N the number of slices, $\vec{v}$ the view direction
1: Load the data $\mathcal{V}(x, y, z) \leftarrow data$
2: create the slices $s_i \in \mathcal{V}$ with each normal $\vec{n_i} \parallel \vec{v}$
3: Turn off the Depth test
4: Enable the Blending
5: **for each** $s_i \in \mathcal{V}$ form back to front **do**
6: $Texture \leftarrow s_i$
7: create a polygon p corresponding to the slice s_i
8: Assign texture coordinates to four corners of the polygon
9: Render and blend the polygon (alpha blending)
10: **end for**

3D textures allow direct treatment of volumetric data. Instead of generating a set of two-dimensional slices in a pre-processing step, the volumetric data is directly downloaded into the graphics hardware, and is directly used to calculate color and opacity values for each pixel covered by a rendered primitive.

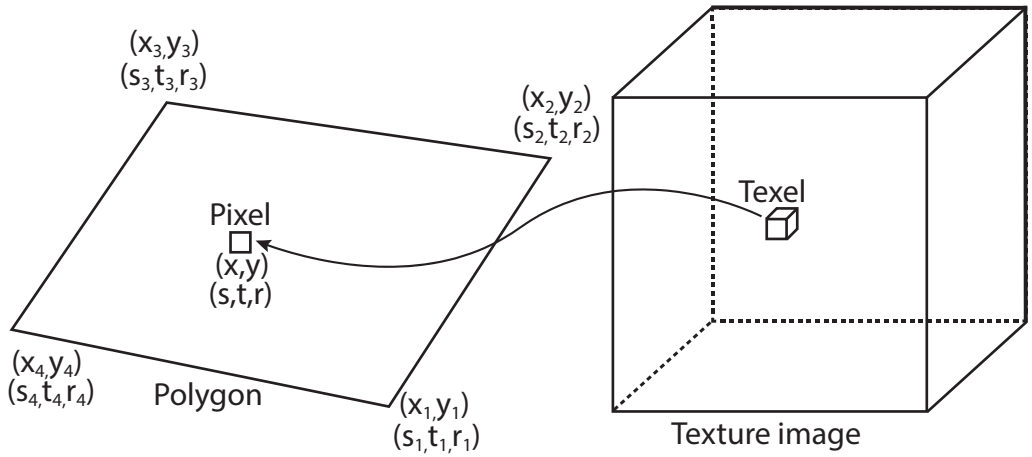


FIGURE 2.3: Calculating color and opacity of a pixel inside a texture-mapped polygon using 3D texture.

To generate oblique slices using 3D texturing, one only has to calculate the vertices of the slice, and then to generate the correct 3D texture coordinates for those vertices, see Figure 2.4. Then these coordinates are passed to the graphics library, and the graphics library will render the projection of the slice onto the image plane, using tri-linear interpolation to reconstruct each pixel's color and opacity values.

Raycasting

Volume raycasting computes a 2D image from the 3D volumetric data set.The basic goal of ray casting is to allow the best use of the three-dimensional data and not attempt to impose any geometric structure on it. It solves one of the most important limitations of surface extraction techniques, namely the way in which they display a projection of a thin shell in the acquisition space. Surface extraction techniques fail

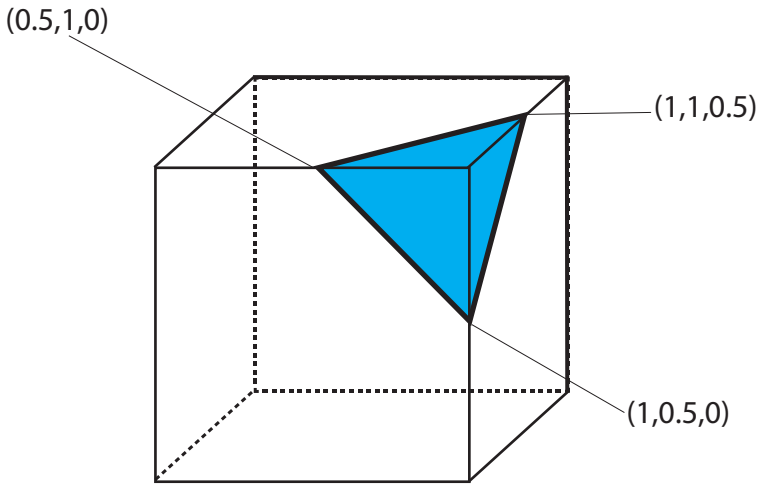


FIGURE 2.4: Calculating vertex coordinates and texture coordinates for an oblique slice. In this figure, the vertex and texture coordinates are identical.

to take into account that, particularly in medical imaging, data may originate from fluid and other materials which may be partially transparent and should be modeled as such. Ray casting doesn't suffer from this limitation. This algorithm can be split in four main steps (see Figure 2.5) :

- **Ray casting**. For each pixel of the final image, a ray of sight is shot ("cast") through the volume. At this stage it is useful to consider the volume being touched and enclosed within a bounding primitive, a simple geometric object (usually a cuboid) that is used to intersect the ray of sight and the volume.
- **Sampling**. Along the part of the ray of sight that lies within the volume, equidistant sampling points or samples are selected. In general, the volume is not aligned with the ray of sight, and sampling points will usually be located in between voxels. Because of that, it is necessary to interpolate the values of the samples from its surrounding voxels (commonly using trilinear interpolation).
- **Shading**. For each sampling point, a transfer function retrieves an RGBA material colour and a gradient of illumination values is computed. The gradient represents the orientation of local surfaces within the volume. The samples are then shaded (i.e. coloured and lit) according to their surface orientation and the location of the light source in the scene.
- **Compositing**. After all sampling points have been shaded, they are composited along the ray of sight, resulting in the final colour value for the pixel that is currently being processed. The composition is derived directly from the rendering equation and is similar to blending acetate sheets on an overhead projector. It may work back-to-front, i.e. computation starts with the sample farthest from the viewer and ends with the one nearest to the viewer. This work flow direction ensures that masked parts of the volume do not affect the resulting pixel. The front-to-back order could be more computationally efficient since, the residual ray energy is getting down while ray travels away from camera; so, the contribution to the rendering integral is diminishing therefore

more aggressive speed/quality compromise may be applied (increasing of distances between samples along ray is one of such speed/quality trade-offs).

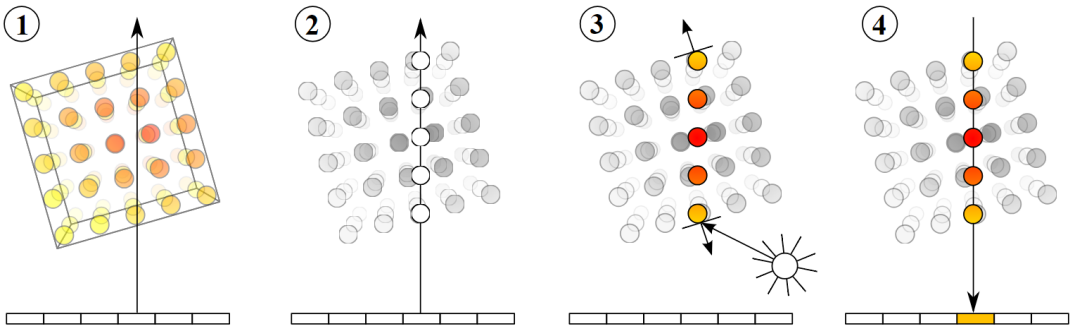


FIGURE 2.5: The four steps of the volume raycasting algorithm

2.1.2 Surface-fitting

2.2 Occlusion management strategies

2.2.1 Transfer Function

A transfer function (TF) maps volumetric data to optical properties and is part of the traditional visualization pipeline: data acquisition, processing, visual mapping, and rendering. Volumetric data is considered to be a scalar function from a three-dimensional spatial domain with a one-dimensional range (e.g., density, flow magnitude, etc.). Image generation involves mapping data samples through the TF, where they are given optical properties such as color and opacity, and compositing them into the image. A TF simultaneously defines which parts of the data are essential to depict and how to depict these, often small, portions of the volumetric data. Considering the first step, a TF is a special, but important, case of a segmentation or classification. With classifi- cation, certain regions in a three-dimensional domain are identified to belong to the same material, such as bone, vessel, or soft tissue, in medical imaging. A plethora of classification and segmentation algorithms have been developed over the last decades, with semiautomatic approaches often tailored to specific application scenarios. Segmentation algorithms can be quite intricate since information from the three-dimensional spatial domain and the one-dimensional data range are taken into account. TFs in their basic form are, on the other hand, restricted to using only the data ranges. In comparison with general classification algorithms, this characteristic makes a TF less powerful with respect to identifying relevant parts of the data. The advantage of a TF is, however, a substantial performance gain as classification based on the one-dimensional data range reduces the complexity tremendously. This gain is the result of the three-dimensional domain being typically two orders of magnitude larger than the small data range. Histograms are an example of discarding potentially complex spatial information and aggregating only the data values to binned-frequency information. In the same spirit, TFs classify interesting parts of the data by considering data values alone. The second functionality of a

TF deals with specifying optical properties for portions of the data range previously identified as being relevant

The transfer functions can be classified according to different criteria. Some of the most important are:

- **The dimension:** The one-dimensional TF classifies the scalar data value, d, and subsequently maps the material to an optical property for rendering: $\mathbf{q}(d) = \mathbf{V}(\mathbf{M}(d))$ where $\mathbf{M}(.)$ is the material classification function and $\mathbf{V}(.)$ is the visual mapping of the material. One-dimensional TFs are adequate in many cases of simulation data where measurement noise is low or even non-existent where different materials of interest have few overlapping intensity ranges Li et al., 2007. One-dimensional TFs are often the first tool available in software packages providing volume rendering, as they are relatively easy to comprehend for the novice or occasional user. Practically all production visualization software, such as ParaView by Ayachit, 2015, VisIt by Childs et al., 2011, or ImageVis3D by Fogal and Krüger, 2010, support 1D TF editors.

  Multidimensional TFs are used for Multidimensional data. If the user has to manipulate the TF definition directly, moving beyond 2D TFs immediately poses significant challenges in terms of user interfaces and cognitive comprehension (Kniss, Kindlmann, and Hansen, 2002). Much research and work on Multidimensional TFs is, therefore, related to various forms of automation and advanced user interfaces. Typical approaches include dimensional reduction, clustering and grouping, machine learning, and various user interface approaches such as parallel coordinates by Guo, Xiao, and Yuan, 2011 or direct slice or volume view interaction.

- **The automation**: The automation includes techniques such as adapting presets, semi-automatic, automatic, and supervised machine learning. Volume rendering has established itself as a powerful visualization tool in many domains, but the necessary design of TFs is a time consuming and tedious task. Consequently, a significant amount of work is dedicated to the automatic and semiautomatic generation of TFs.

- **The aggregated attributes** : Conceptually, the introduction of additional quantities in the TF definition makes it possible to discriminate more parts in a dataset. Histogram clustering helps to reduce the degrees of freedom in designing a TF. As an illustration, Wang et al., 2011 propose modeling the histogram space using a Gaussian mixture model. An elliptical TF is assigned to each Gaussian, and the user interaction is simplified in order to edit the parameters of these ellipsoids.

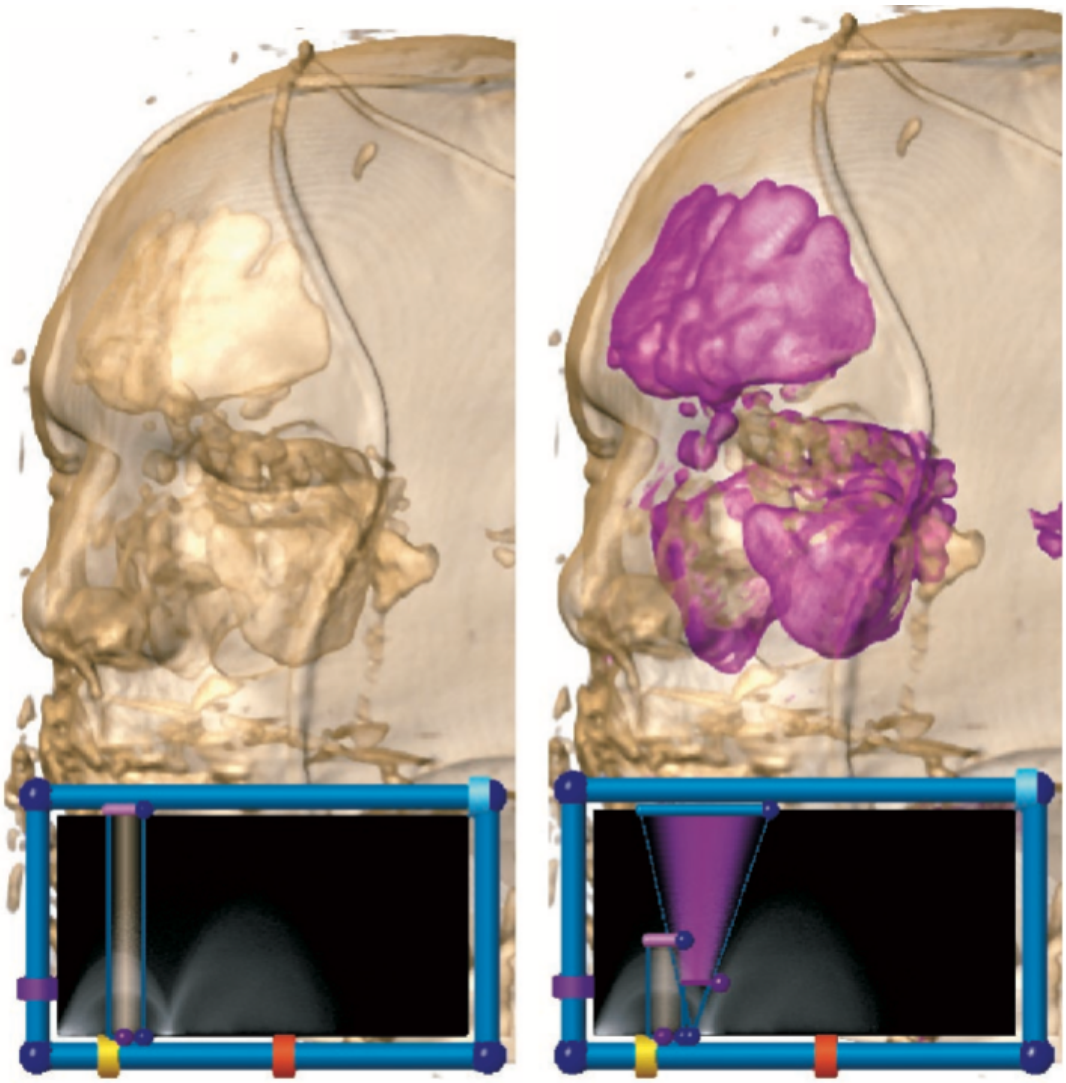


FIGURE 2.6: The two-dimensional TF editor widgets allow the user to more precisely control the classification power of the 2D TF. The triangular widget can also be skewed to better fit the desired classification domain (Kniss, Kindlmann, and Hansen, 2002)

2.2.2 Segmentation

2.2.3 Deformations and Focus + Context

Almost all images used in visualization are computed with a planar pinhole camera (PPC). One reason for this is that the PPC models the human eye well, producing images that resemble what users would see if they were actually looking at the dataset to be visualized. Another reason is simplicity: software and hardware implementations of PPC rendering algorithms allow visualizing complex 3-D datasets at interactive rates. However, the simplicity of the PPC model also brings three important limitations. First, the PPC has a limited field of view. Second, the sampling rate of the PPC is uniform over its entire field of view. Third, the PPC has only a single viewpoint, i.e. the pinhole where all rays converge.

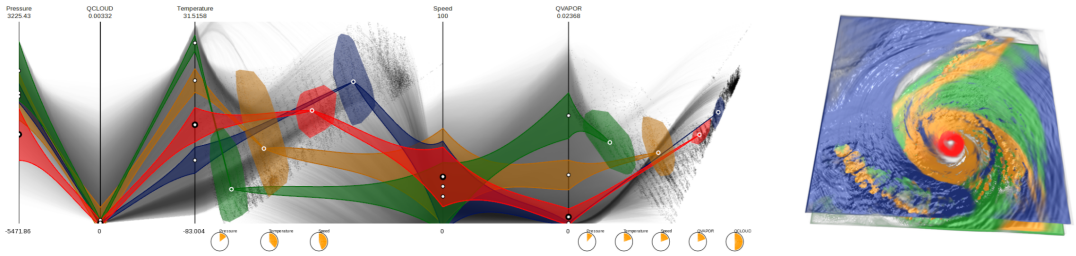


FIGURE 2.7: Parallel coordinates with embedded multidimensional scale plots for editing multidimensional TFs(Guo, Xiao, and Yuan, 2011)

Multiperspective visualization is a promising approach based on integrating data sampled from multiple viewpoints into a single image. The multiple viewpoints are integrated tightly which alleviates the visualization discontinuity problem of multiple individual images. Like in the case of the dataset distortion approach, multiperspective visualization amounts to a warp of global spatial relationships between data subsets. However, multiperspective visualization allows specifying the desired disocclusion effect directly in the image, as opposed to indirectly, through a dataset distortion. Finally, multiperspective visualization does not preclude but rather enhances interactive exploration of datasets. The multiperspective image provides a preview of data subsets to come which improves interactive visualization efficiency. Multiperspective visualization has challenges of its own. The multiperspective image is computed using a non-pinhole camera model that does not project 3-D lines to image plane lines, so one challenge is achieving the desired disocclusion effect while minimizing visualization distortions. The non-pinhole camera model is considerably more complex than the PPC model, so a second challenge is to achieve adequate rendering performance to support interactive visualization and dynamic datasets. Eliminating the single viewpoint constraint of the PPC model results in a multidimensional camera model design space. Whereas for the PPC model the only intrinsic parameter of significant relevance in shaping the visualization is the focal length, optimizing multiperspective visualization requires tuning a large number of parameters. Consequently a third challenge is to specify the camera model that best visualizes a given dataset from a given location.
Cui et al., 2010 proposed a multiperspective visualization technique based on the curved ray camera (CRC). The CRC is designed to address the challenges of multiperspective visualization described above. The CRC integrates multiple viewpoints seamlessly. The curved rays allow for a progressive transition between one viewpoint and the next. A CRC ray is a sequence of line segments connected by conic curve segments. Each conic connects consecutive line segments with C' continuity, which alleviates visualization distortions. The CRC provides a fast projection operation which allows rendering 3-D surface datasets efficiently by projection followed by rasterization, with the help of graphics hardware. The rays of the CRC can be traced inexpensively which enables visualization techniques that require ray casting, such as volume rendering.
Wu and Popescu, 2016 extended this work by proposing a framework that allows constructing continuous multiperspective visualizations by changing the viewpoint for individual focus regions in an image, by connecting input images with continuous context, and by alleviating occlusions to moving targets without distorting or displacing the target subimages.

An interactive lens is a lightweight tool to solve a localized visualization problems by temporarily altering a selected part of the visual representation of the data Tominski et al., 2016. Using this lens approach, we propose an interactive volume deformation based on GPU accelerated ray-casting to free a designated target from local occlusion while keeping the global context.

A lens is a parameterizable selection according to which a base visualization is altered. Typically, a lens is added to a visualization to interactively solve a specific localized problem. This property is very interesting with the aim of providing a focus+context solution to occlusion in volume rendering. Lenses can have different geometric properties not only defining their appearance, but also determining the selection, which is the subset of the data where these lenses take effect. The major geometric properties of a lens are shape, position, and size as well as the orientation. The shape of a lens is usually chosen to fulfill the requirements of the application and is strongly linked to the lens function. Most virtual lenses are circular Tominski et al., 2006 or are rectangular Kincaid, 2010 as the real-world ones (magnifying glass, windows). Our lens has also a circular shape in order to remind its magnifying property. Some lenses, such as the JellyLens Pindat et al., 2012 and the smart lenses Thiede, Fuchs, and Schumann, 2008 can adapt their shape automatically according to the data. The position and the size parametrization can increase the flexibility of an interactive lens. Modifying this position or size will set its focus on a different part of the data according to the user's interest. It is possible to update automatically these parameters in order to guide the user toward interesting events in the data Tominski, 2011, or adjust the lens position according to predefined paths as the Routelens Alvina et al., 2014 does. With this mind, our lens updates automatically its properties once a target has been selected. This allows a smooth transition towards an unobstructed and magnified area of interest.

Lenses for volume visualization face challenges mainly related to spatial selection and occlusion. Wang et al., 2005 addressed these issues by proposing the Magic Lens . This framework propose FOUR different types of lenses allowing a free-form volumetric lens function that can be feature-adaptive or user-configurable for a high-quality aliased, and interactive display with smooth transitions from highto low-resolution areas.

In addition to interactively magnifying areas and objects of interest, our lens frees them from obstruction and allows local modification of the camera to see the target under other perspectives. Tong, Li, and Shen, 2017 proposed the GlyphLens that removes the occluding glyphs by pulling the glyphs aside through the animation, but this tool is only well suited for systems where 3D volumetric dataset are visualized using glyphs. Lenses can create discontinuities between their inner part and the rest of the volume. Deformation can be a solution to this discontinuity issue.

Hsu, Ma, and Correa, 2011 developed a framework that can generate non-linear sampling rays that smoothly project objects in a scene at multiple levels of detail onto a single image. Such a technique requires a lot of computational time to render a single image from features of interest at different scales, see figure 2.8.This multiscale rendering framework consists of a sequence of pinhole cameras, each of which captures a view of interest at a particular scale. The camera rays for each pixel in the final projected image are generated based on a user-defined image mask

which specifies the interesting regions in each view. The camera rays non-linearly cast through all scales of interest. Since the camera rays in the model are bent coherently and march consistently, the objects projected on the image are continuous in both image-space and object-space. Their method starts by setting up several pinhole cameras for viewing different scales of interest, utilizing most users' ease and familiarity with manipulating ordinary pinhole cameras. Each camera produces an image of its view. In order to combine all such views to form a single multiscale image, a user-specified image mask is used to indicate regions of each view that the user would like to show in the final image. In other words, every camera projects only part of its view onto the final image space, based on a corresponding image mask. In order to ensure consistency while projecting different camera viewpoints onto different parts of the image, we force all camera rays to originate from the first camera, which is the one that has the largest scale of view. The rest of the cameras define intermediate points that camera rays must pass through in order, from the largest to smallest scales. They use Bézier curves to connect the nearclipping planes of two successive cameras.

The rays are bent gradually from one scale to another to maintain object-space continuity as well as image-space smoothness. The multiscale camera can also achieve a focus+context effect, a technique frequently used in many visualization applications. Finally, we show that it can potentially create pictures that mimic artistic or impossible views of objects or scenes like the kind made by the artist M. C. Escher.

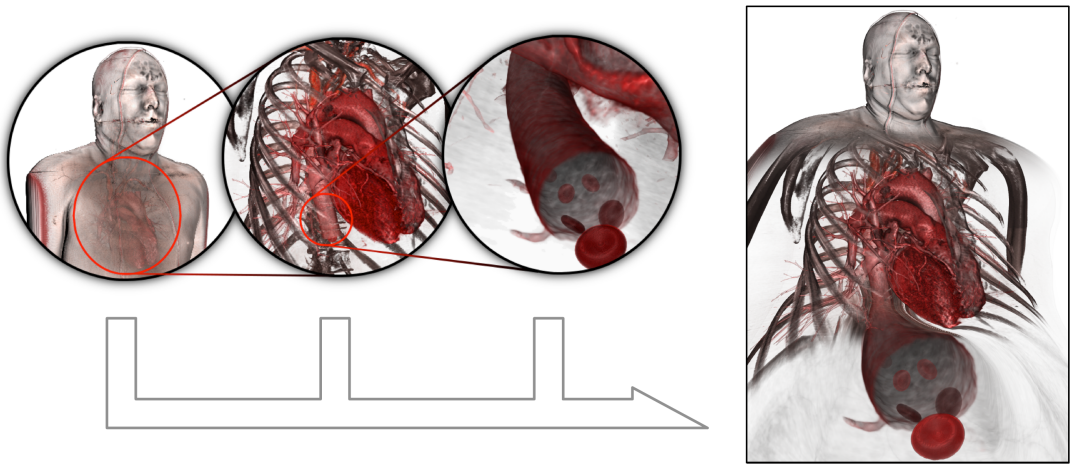


FIGURE 2.8: A Rendering Framework for Multiscale Views of 3D Models. A continuous multiscale view (right) of a volumetric human body dataset shows three different levels of detail (left three) in a single image. The image on the right is directly rendered with our multiscale framework.

Bruckner and Groller Bruckner and Groller, 2006 proposed exploded view for volume data by partitioning the volume into several segments (figure 2.9). Basically, in an exploded view the object is decomposed into several parts which are displaced so that internal details are visible. This does not only give an unobstructed view on the focus but also potentially reveals other interesting information, such as cross-sections of the split object. The advantage of exploded views is that they simultaneously convey the global structure of the depicted object, the details of individual

components, and the local relationships among them. A Force-directed layout approach is used to arrange three-dimensional objects in such a way that they do not occlude another object, but with as little displacement as possible. Four forces are used. **The Return force** is an attractive force that tries to move the parts towards their original location. The force F_r is realized as a logarithmic spring:

$$F_r = c_r \ln(||r||) \cdot \frac{r}{||r||}$$

where r is the vector from the vertex's current position to its original location and c_r is a constant factor. **The Explosion force** drives the specified parts away from our selection object. The explosion point is also weighted according to the size of the region corresponding to an octree node. Each explosion point exerts a force F_e on every part P_i:

$$F_e = \frac{c_e}{e^{||r||}} \cdot \frac{r}{||r||}$$

where r is the vector from the explosion point to the closest point of the part geometry of P_i and c_e is a scaling factor. **The Viewing force** is a view-dependent force which attempts to arrange parts so that they do not occlude the selection for the current viewing transformation. The force F_v is:

$$F_v = \frac{c_v}{||r||} \cdot \frac{r}{||r||}$$

where r is the vector from the closest point along the viewing ray to the center of the body and c_v is a scaling factor. **The Spacing force** prevents clustering of parts, we also add a repulsive force F_s. For a part P_i, the spacing force exerted by another part P_j is:

$$F_s = \frac{c_s}{||r||^2} \cdot \frac{r}{||r||}$$

where r is the vector from the center of P_j to the center of P_i and c_s is a constant scaling factor.

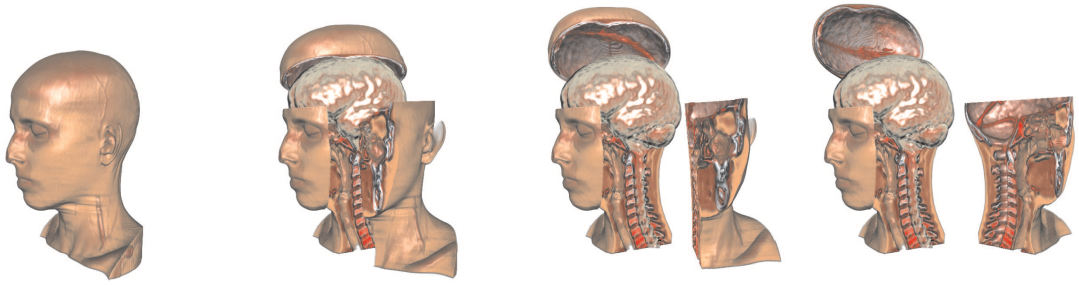


FIGURE 2.9: Interactive exploded-view illustration of a human head with increasing degrees-of-explosion. Two hinge joints are used to constrain part movement

McGuffin, Tancau, and Balakrishnan, 2003 performed deformations using peeling to see hidden parts of the data. Their goal in using deformations is to increase the visibility of the inner portions of the volume, without completely removing the surrounding data that normally occludes the inside. This is akin to focus+context schemes that allow a user to zoom-in on data of interest, while using remaining

screen space to show the surrounding context. Appropriately chosen deformations could, for example, split open a volume, showing displaced structures side-by-side, making it easy for the user to see how they connect, and allowing the user to mentally stitch them together into a whole. Deformations with familiar real-world analogues (e.g. cutting and peeling the skin off a fruit, or the layers off an onion) are also likely to be readily understood by users. Therefore in their work, they describe a prototype system that implements different metaphors for deformation-based browsing of volumetric data. As will be seen, a key element of our approach is to support differential treatment of the various semantic layers in a data set. The term "semantic layers", means subsets of the data that are useful or meaningful to the user. These layers could be defined geometrically, for example as sections created with parallel planar cuts. More typically, layers would depend on the voxel data values, for example boundaries that are found during segmentation or isosurface extraction. In the context of medical visualization, there is at least anecdotal evidence that anatomists, for example, prefer to remove tissue layer by layer, rather than making arbitrary planar cuts.

Deformations can reveal predefined features in the dataset by taking into account the precomputed segmentation. Tong et al. proposed a deforming Lens which moves streamlines to observe the inner part of streamline bundles Tong et al., 2016.

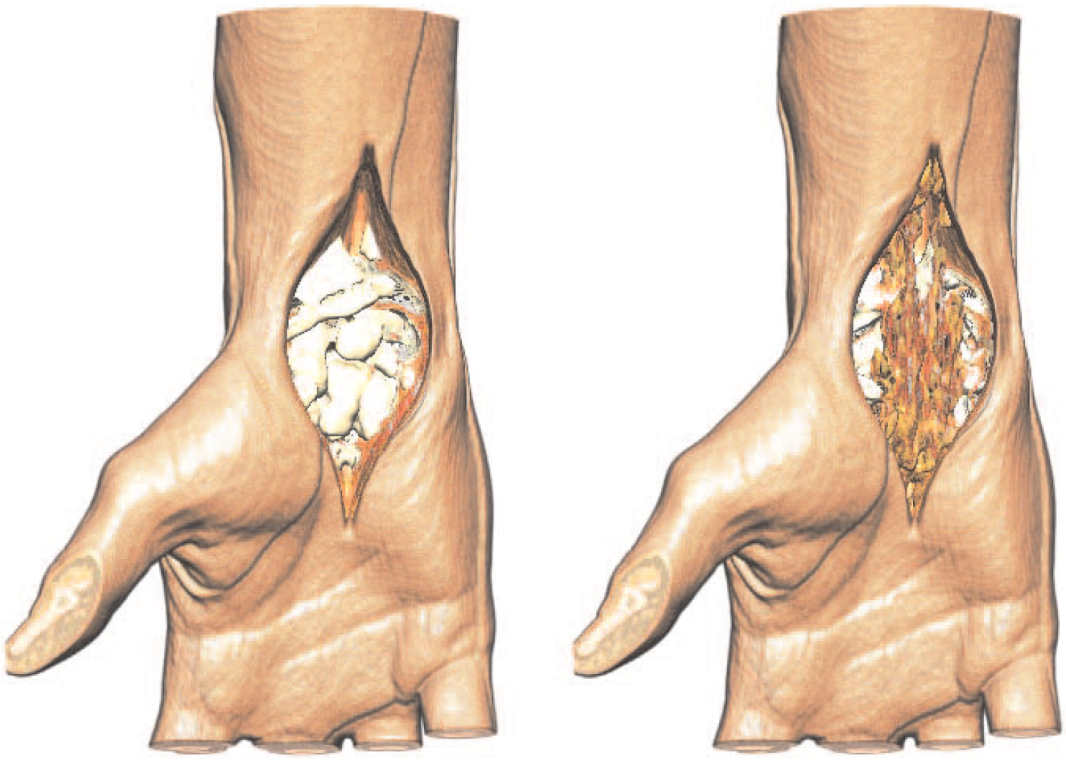
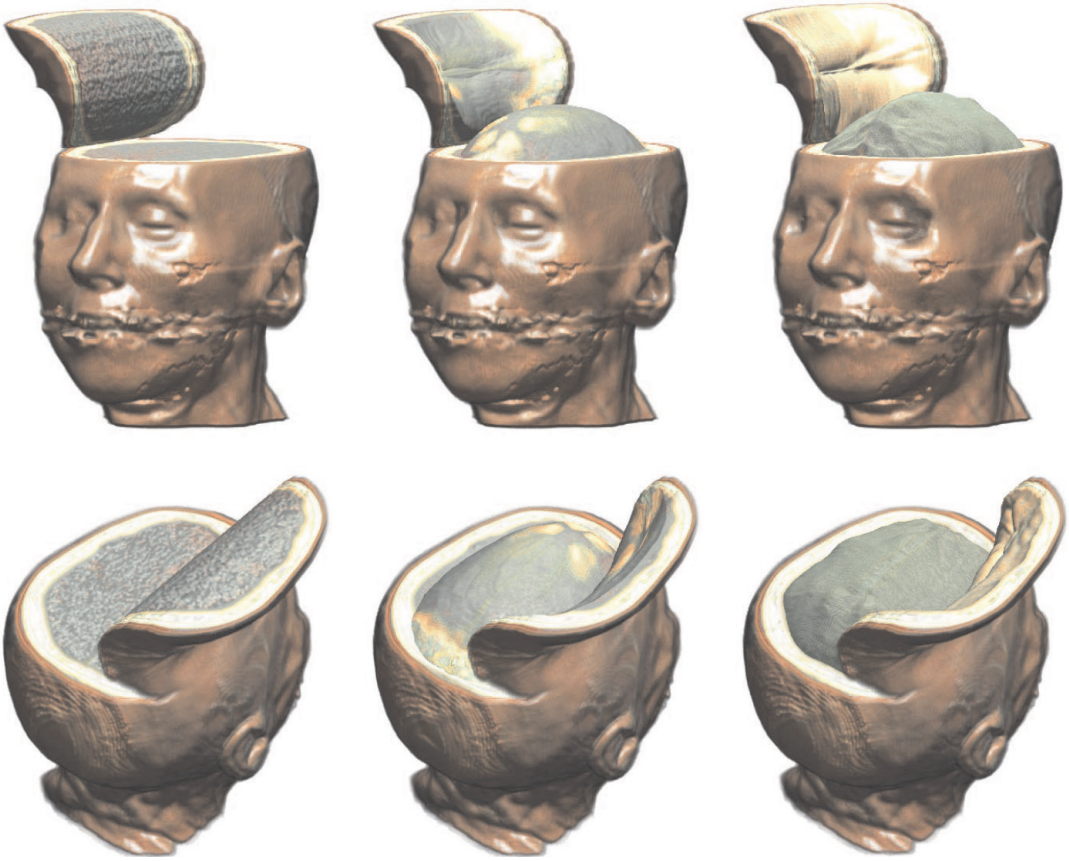


FIGURE 2.10: A feature-aligned retraction applied to a human hand data set, showing bones (left) and vessels (right)

Correa, Silver, and Chen, 2007 proposed a framework allowing the users to physically or actively manipulate the geometry of a data object. Some studies performed deformations using surgical metaphors like Islam, Silver, and Chen, 2007; Correa, Silver, and Chen, 2006 to see hidden parts of the volume, see figure 2.10. They

propose a feature-based technique for manipulating volumetric objects for illustrative visualization, and describe a GPU-based implementation that enables interactive specification and their rendering. Inspired by medical illustrations that frequently depict the results of manipulation with tools such as peelers, retractors and pliers, this system allows the specification of manipulation through a collection of procedurally-defined manipulation operators.



(a) Axis Alignment (b) Surface Alignment (c) Segment Alignment

FIGURE 2.11: An example of different types of alignment. (a) Axis aligned peel. Note how the peeled layer is thick and flat, since it is aligned with an orthogonal axis. (b) Surface aligned peel, aligned with a computed distance field. Notice how it approximates a surface. (c) Segment aligned peel, based on segmentation, which is more accurate. Note that in the feature based alignment (b) and (c) the peel is thin and rounded.

Correa et al. introduced two feature-based methods, namely surface alignment and segment alignment for modeling and rendering illustrative manipulation, and compare them with the traditional axis alignment method, see figure 2.11. They devise a method for estimating accurate normals along the surface of cuts and dissections while maintaining continuous normals, which allows us to obtain correct shading of the object being manipulated, opaque or translucent. The four operators proposed in this framework are the following:

- Peeler: The peeler simulates peeling or cutting of the outer layers of a volumetric model. Different parameters of the peeler, such as thickness of the cut and the angle of bending of the peeled layer, can be manipulated interactively to obtain different illustrative effects.
- Retractors: In the context of surgical illustration, retractors are tools used to spread organs or bones, or to hold back soft tissues such as skin. In the context of illustration, they are useful for illustrating the access to the internal organs.
- Pliers: Pliers are tools used to grasp tissue and pull it or poke it into the volumetric object. The operator parameters include the shape of the displacement and the radius of influence which specifies how the displacement propagates through the volume.
- Dilator: Dilators are used to gain access into narrow regions, cuts or vessels. They essentially scale the region from the inside typically by blowing air, to increase the visibility or accessibility of the region. When applied to volumetric manipulation, dilators have a similar effect to that of volumetric magic lenses (Wang et al., 2005).

However this framework do not offer tools for local manipulation and exploration inside the cut, which allows perceiving a target under a different perspective while keeping the global context.

Hurter et al., 2014 propose a set of linked views that display subsets of data attributes. Example views are 3D DVR plots, 2D scatterplots, and 2D and 3D histograms. Views are linked by brushing and free-form selection. Using the optimal view(s), one can find structures of interest, e.g. spatially compact zones in DVR renderings or peaks in histograms, and highlight or erase such structures in all views at once. They enhance classical histogram views with shading, sorting, and depth, to allow spotting complex patterns with greater ease. In addition, they propose a smooth animation between multiple views, to locate (and select) data patterns which are hard to isolate in static plots. This framework integrate a Focus+Context deformation that adds the ability to uncover locally occluded spatial patterns in different views. The GPU implementation of their techniques, which they call color tunneling, creates real-time interactive, animated, explorations of datasets of tens of millions of points on a modern PC. Since the goal is to explore dense multivariate datasets which exhibit significant overlap between groups of voxels or pixels, tools to interactively unveil the occluded structures of interest are needed. Five interactive techniques have been proposed as follows (see figure 2.12):

- View linking: Three configurable views (2 exploration views and one lock view) for data exploration,
- Warp: Animates a view between two configurations,
- Lock: Locks items not to be affected by brush or dig ,
- Brush: With the lock view, brushing allows adding or removing data in a view,
- Dig: With the lock view, digging pushes data points away from the lens center to unveil occluded structures

2.3 Volume rendering in Virtual Reality and Augmented Reality

2.3.1 Remote volume rendering

Volumetric data exploration with direct volume rendering technique is of great help to visually extract relevant structures in many fields of science: medical imaging **ljung_full_2006**, astrophysics and more recently in baggage inspection. To leverage this knowledge extraction, many techniques have been developed.

In this section, we detail existing ones with volume visualization, transfer function, direct voxel manipulation, and focus plus context interaction. Volume visualization can be done with geometric rendering system which transforms the data into a set of polygons representing an iso-surface. The contour tree algorithm **carr_computing_2000** and other alternatives such as branch decomposition **pascucci_multi-resolution_2004** are usually used to find these iso-surfaces. According to H. Guo **guo_local_2013**, contour trees algorithms are vulnerable to noises, which can be problematic in baggage inspections since dense materials (e.g. iron) cause noises by reflecting the X-rays. For this reason we used the volume rendering technique; Chen et al. provide a review of existing techniques **chen_3-d_2000**. In order to investigate volumetric dataset, one can use the Transfer Function (TF). In practice, it maps the voxel density with a specific color (including its transparency). Transfer function can be 1D, 2D or nD and are of great help to isolate structures of interest in volumetric data **kniss_multidimensional_2002**. Thanks to the color blending process, suitable transfer function can also reveal iso surface or hide density to improve the volumetric data visualization. The setup of this transfer function remains complex, but some automatic systems provide solution thanks to data analysis **correa_size-based_2008** **sereda_visualization_2006** **patel_moment_2009** or user interactions **guo_wysiwyg_2011**. Since our users have a limited knowledge on volume rendering, we developed new interaction techniques with sufficient abstraction level regarding technical constraints. These techniques will be detailed in this paper. As an example, we investigated predefined transfer functions with smooth transitions when changing their setup to reduce disruptive animation effects **tversky_animation:_2002**. Since graphic card power never stops to improve, new techniques allow the direct manipulation of the voxels which composes the volume to be displayed. Color tunneling introduced a set of interactions (lock, Brush, Dig) to explore 2D and 3D dataset composed of pixels of voxels **hurter_interactive_2014** with point based rendering techniques **sainz_point-based_2004** . Histomage also provides direct manipulation of pixels thanks to a histogram which can be used as a new selection and pixel modification tool **chevalier_histomages:_2012**. Other interactive systems investigated pixel exploration with a lens as a focus plus context technique **elmqvist_color_2011** **hurter_moleview:_2011**. Our work provides additional interaction techniques with pixel based techniques **hurter_interactive_2014** to address occlusion issues, focus and context awareness, reversibility of the actions, and continuity.

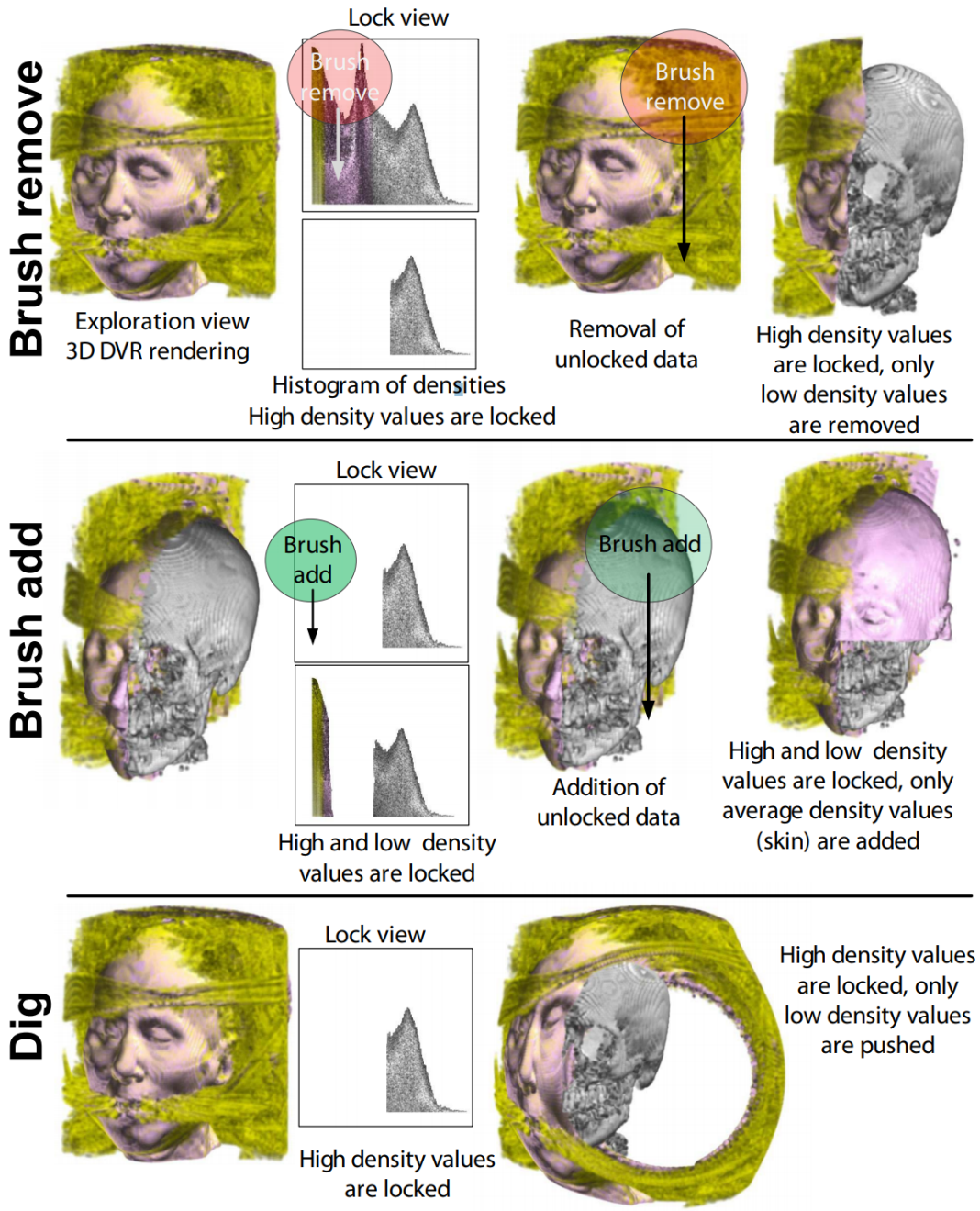


FIGURE 2.12: Brush, dig, and lock tools. Locked items are not affected by brush and dig. We used transfer functions to make air voxels transparent and standard gradient shading. We see that the head is surrounded by a large amount of uninteresting noise (yellow).

Chapter 3

Chapter Title Here

3.1 Welcome and Thank You

Welcome to this LaTeX Thesis Template, a beautiful and easy to use template for writing a thesis using the LaTeX typesetting system.

If you are writing a thesis (or will be in the future) and its subject is technical or mathematical (though it doesn't have to be), then creating it in LaTeX is highly recommended as a way to make sure you can just get down to the essential writing without having to worry over formatting or wasting time arguing with your word processor.

LaTeX is easily able to professionally typeset documents that run to hundreds or thousands of pages long. With simple mark-up commands, it automatically sets out the table of contents, margins, page headers and footers and keeps the formatting consistent and beautiful. One of its main strengths is the way it can easily typeset mathematics, even *heavy* mathematics. Even if those equations are the most horribly twisted and most difficult mathematical problems that can only be solved on a super-computer, you can at least count on LaTeX to make them look stunning.

3.2 Learning LaTeX

LaTeX is not a WYSIWYG (What You See is What You Get) program, unlike word processors such as Microsoft Word or Apple's Pages. Instead, a document written for LaTeX is actually a simple, plain text file that contains *no formatting*. You tell LaTeX how you want the formatting in the finished document by writing in simple commands amongst the text, for example, if I want to use *italic text for emphasis*, I write the `\emph{text}` command and put the text I want in italics in between the curly braces. This means that LaTeX is a "mark-up" language, very much like HTML.

3.2.1 A (not so short) Introduction to LaTeX

If you are new to LaTeX, there is a very good eBook – freely available online as a PDF file – called, "The Not So Short Introduction to LaTeX". The book's title is typically shortened to just *lshort*. You can download the latest version (as it is occasionally updated) from here: http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf

It is also available in several other languages. Find yours from the list on this page: http://www.ctan.org/tex-archive/info/lshort/

It is recommended to take a little time out to learn how to use LaTeX by creating several, small 'test' documents, or having a close look at several templates on: http://www.LaTeXTemplates.com
Making the effort now means you're not stuck learning the system when what you *really* need to be doing is writing your thesis.

3.2.2 A Short Math Guide for LaTeX

If you are writing a technical or mathematical thesis, then you may want to read the document by the AMS (American Mathematical Society) called, "A Short Math Guide for LaTeX". It can be found online here: http://www.ams.org/tex/amslatex.html under the "Additional Documentation" section towards the bottom of the page.

3.2.3 Common LaTeX Math Symbols

There are a multitude of mathematical symbols available for LaTeX and it would take a great effort to learn the commands for them all. The most common ones you are likely to use are shown on this page: http://www.sunilpatel.co.uk/latex-type/latex-math-symbols/

You can use this page as a reference or crib sheet, the symbols are rendered as large, high quality images so you can quickly find the LaTeX command for the symbol you need.

3.2.4 LaTeX on a Mac

The LaTeX distribution is available for many systems including Windows, Linux and Mac OS X. The package for OS X is called MacTeX and it contains all the applications you need – bundled together and pre-customized – for a fully working LaTeX environment and work flow.

MacTeX includes a custom dedicated LaTeX editor called TeXShop for writing your '.tex' files and BibDesk: a program to manage your references and create your bibliography section just as easily as managing songs and creating playlists in iTunes.

3.3 Getting Started with this Template

If you are familiar with LaTeX, then you should explore the directory structure of the template and then proceed to place your own information into the *THESIS INFORMATION* block of the `main.tex` file. You can then modify the rest of this file to your unique specifications based on your degree/university. Section 3.5 on page 26 will help you do this. Make sure you also read section 3.7 about thesis conventions to get the most out of this template.

If you are new to LaTeX it is recommended that you carry on reading through the rest of the information in this document.

Before you begin using this template you should ensure that its style complies with the thesis style guidelines imposed by your institution. In most cases this template style and layout will be suitable. If it is not, it may only require a small change to bring the template in line with your institution's recommendations. These modifications will need to be done on the `MastersDoctoralThesis.cls` file.

3.3.1 About this Template

This LaTeX Thesis Template is originally based and created around a LaTeX style file created by Steve R. Gunn from the University of Southampton (UK), department of Electronics and Computer Science. You can find his original thesis style file at his site, here: `http://www.ecs.soton.ac.uk/~srg/softwaretools/document/templates/`

Steve's `ecsthesis.cls` was then taken by Sunil Patel who modified it by creating a skeleton framework and folder structure to place the thesis files in. The resulting template can be found on Sunil's site here: `http://www.sunilpatel.co.uk/thesis-template`

Sunil's template was made available through `http://www.LaTeXTemplates.com` where it was modified many times based on user requests and questions. Version 2.0 and onwards of this template represents a major modification to Sunil's template and is, in fact, hardly recognisable. The work to make version 2.0 possible was carried out by Vel and Johannes Böttcher.

3.4 What this Template Includes

3.4.1 Folders

This template comes as a single zip file that expands out to several files and folders. The folder names are mostly self-explanatory:

Appendices – this is the folder where you put the appendices. Each appendix should go into its own separate `.tex` file. An example and template are included in the directory.

Chapters – this is the folder where you put the thesis chapters. A thesis usually has about six chapters, though there is no hard rule on this. Each chapter should go in its own separate `.tex` file and they can be split as:

- Chapter 1: Introduction to the thesis topic
- Chapter 2: Background information and theory
- Chapter 3: (Laboratory) experimental setup
- Chapter 4: Details of experiment 1
- Chapter 5: Details of experiment 2
- Chapter 6: Discussion of the experimental results
- Chapter 7: Conclusion and future directions

This chapter layout is specialised for the experimental sciences, your discipline may be different.

Figures – this folder contains all figures for the thesis. These are the final images that will go into the thesis document.

3.4.2 Files

Included are also several files, most of them are plain text and you can see their contents in a text editor. After initial compilation, you will see that more auxiliary files are created by LaTeX or BibTeX and which you don't need to delete or worry about:

example.bib – this is an important file that contains all the bibliographic information and references that you will be citing in the thesis for use with BibTeX. You can write it manually, but there are reference manager programs available that will create and manage it for you. Bibliographies in LaTeX are a large subject and you

may need to read about BibTeX before starting with this. Many modern reference managers will allow you to export your references in BibTeX format which greatly eases the amount of work you have to do.

MastersDoctoralThesis.cls – this is an important file. It is the class file that tells LaTeX how to format the thesis.

main.pdf – this is your beautifully typeset thesis (in the PDF file format) created by LaTeX. It is supplied in the PDF with the template and after you compile the template you should get an identical version.

main.tex – this is an important file. This is the file that you tell LaTeX to compile to produce your thesis as a PDF file. It contains the framework and constructs that tell LaTeX how to layout the thesis. It is heavily commented so you can read exactly what each line of code does and why it is there. After you put your own information into the *THESIS INFORMATION* block – you have now started your thesis!

Files that are *not* included, but are created by LaTeX as auxiliary files include:

main.aux – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main `.tex` file.

main.bbl – this is an auxiliary file generated by BibTeX, if it is deleted, BibTeX simply regenerates it when you run the `main.aux` file. Whereas the `.bib` file contains all the references you have, this `.bbl` file contains the references you have actually cited in the thesis and is used to build the bibliography section of the thesis.

main.blg – this is an auxiliary file generated by BibTeX, if it is deleted BibTeX simply regenerates it when you run the main `.aux` file.

main.lof – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main `.tex` file. It tells LaTeX how to build the *List of Figures* section.

main.log – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main `.tex` file. It contains messages from LaTeX, if you receive errors and warnings from LaTeX, they will be in this `.log` file.

main.lot – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main `.tex` file. It tells LaTeX how to build the *List of Tables* section.

main.out – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main `.tex` file.

So from this long list, only the files with the `.bib`, `.cls` and `.tex` extensions are the most important ones. The other auxiliary files can be ignored or deleted as LaTeX and BibTeX will regenerate them.

3.5 Filling in Your Information in the `main.tex` File

You will need to personalise the thesis template and make it your own by filling in your own information. This is done by editing the `main.tex` file in a text editor or your favourite LaTeX environment.

Open the file and scroll down to the third large block titled *THESIS INFORMATION* where you can see the entries for *University Name*, *Department Name*, etc ...

Fill out the information about yourself, your group and institution. You can also insert web links, if you do, make sure you use the full URL, including the `http://` for this. If you don't want these to be linked, simply remove the `\href{url}{name}` and only leave the name.

When you have done this, save the file and recompile `main.tex`. All the information you filled in should now be in the PDF, complete with web links. You can now begin your thesis proper!

3.6 The `main.tex` File Explained

The `main.tex` file contains the structure of the thesis. There are plenty of written comments that explain what pages, sections and formatting the LaTeX code is creating. Each major document element is divided into commented blocks with titles in all capitals to make it obvious what the following bit of code is doing. Initially there seems to be a lot of LaTeX code, but this is all formatting, and it has all been taken care of so you don't have to do it.

Begin by checking that your information on the title page is correct. For the thesis declaration, your institution may insist on something different than the text given. If this is the case, just replace what you see with what is required in the *DECLARATION PAGE* block.

Then comes a page which contains a funny quote. You can put your own, or quote your favourite scientist, author, person, and so on. Make sure to put the name of the person who you took the quote from.

Following this is the abstract page which summarises your work in a condensed way and can almost be used as a standalone document to describe what you have done. The text you write will cause the heading to move up so don't worry about running out of space.

Next come the acknowledgements. On this page, write about all the people who you wish to thank (not forgetting parents, partners and your advisor/supervisor).

The contents pages, list of figures and tables are all taken care of for you and do

not need to be manually created or edited. The next set of pages are more likely to be optional and can be deleted since they are for a more technical thesis: insert a list of abbreviations you have used in the thesis, then a list of the physical constants and numbers you refer to and finally, a list of mathematical symbols used in any formulae. Making the effort to fill these tables means the reader has a one-stop place to refer to instead of searching the internet and references to try and find out what you meant by certain abbreviations or symbols.

The list of symbols is split into the Roman and Greek alphabets. Whereas the abbreviations and symbols ought to be listed in alphabetical order (and this is *not* done automatically for you) the list of physical constants should be grouped into similar themes.

The next page contains a one line dedication. Who will you dedicate your thesis to?

Finally, there is the block where the chapters are included. Uncomment the lines (delete the % character) as you write the chapters. Each chapter should be written in its own file and put into the *Chapters* folder and named `Chapter1`, `Chapter2`, etc...Similarly for the appendices, uncomment the lines as you need them. Each appendix should go into its own file and placed in the *Appendices* folder.

After the preamble, chapters and appendices finally comes the bibliography. The bibliography style (called *authoryear*) is used for the bibliography and is a fully featured style that will even include links to where the referenced paper can be found online. Do not underestimate how grateful your reader will be to find that a reference to a paper is just a click away. Of course, this relies on you putting the URL information into the BibTeX file in the first place.

3.7 Thesis Features and Conventions

To get the best out of this template, there are a few conventions that you may want to follow.

One of the most important (and most difficult) things to keep track of in such a long document as a thesis is consistency. Using certain conventions and ways of doing things (such as using a Todo list) makes the job easier. Of course, all of these are optional and you can adopt your own method.

3.7.1 Printing Format

This thesis template is designed for double sided printing (i.e. content on the front and back of pages) as most theses are printed and bound this way. Switching to one sided printing is as simple as uncommenting the *oneside* option of the `documentclass`

command at the top of the `main.tex` file. You may then wish to adjust the margins to suit specifications from your institution.

The headers for the pages contain the page number on the outer side (so it is easy to flick through to the page you want) and the chapter name on the inner side.

The text is set to 11 point by default with single line spacing, again, you can tune the text size and spacing should you want or need to using the options at the very start of `main.tex`. The spacing can be changed similarly by replacing the *singlespacing* with *onehalfspacing* or *doublespacing*.

3.7.2 Using US Letter Paper

The paper size used in the template is A4, which is the standard size in Europe. If you are using this thesis template elsewhere and particularly in the United States, then you may have to change the A4 paper size to the US Letter size. This can be done in the margins settings section in `main.tex`.

Due to the differences in the paper size, the resulting margins may be different to what you like or require (as it is common for institutions to dictate certain margin sizes). If this is the case, then the margin sizes can be tweaked by modifying the values in the same block as where you set the paper size. Now your document should be set up for US Letter paper size with suitable margins.

3.7.3 References

The `biblatex` package is used to format the bibliography and inserts references such as this one (Hawthorn, Weber, and Scholten, 2001). The options used in the `main.tex` file mean that the in-text citations of references are formatted with the author(s) listed with the date of the publication. Multiple references are separated by semicolons (e.g. (Wieman and Hollberg, 1991; Hawthorn, Weber, and Scholten, 2001)) and references with more than three authors only show the first author with *et al.* indicating there are more authors (e.g. (Arnold et al., 1998)). This is done automatically for you. To see how you use references, have a look at the `Chapter1.tex` source file. Many reference managers allow you to simply drag the reference into the document as you type.

Scientific references should come *before* the punctuation mark if there is one (such as a comma or period). The same goes for footnotes[1]. You can change this but the most important thing is to keep the convention consistent throughout the thesis. Footnotes themselves should be full, descriptive sentences (beginning with a capital letter and ending with a full stop). The APA6 states: "Footnote numbers should be superscripted, [...], following any punctuation mark except a dash." The Chicago manual of style states: "A note number should be placed at the end of a sentence

[1] Such as this footnote, here down at the bottom of the page.

or clause. The number follows any punctuation mark except the dash, which it precedes. It follows a closing parenthesis."

The bibliography is typeset with references listed in alphabetical order by the first author's last name. This is similar to the APA referencing style. To see how LaTeX typesets the bibliography, have a look at the very end of this document (or just click on the reference number links in in-text citations).

A Note on bibtex

The bibtex backend used in the template by default does not correctly handle unicode character encoding (i.e. "international" characters). You may see a warning about this in the compilation log and, if your references contain unicode characters, they may not show up correctly or at all. The solution to this is to use the biber backend instead of the outdated bibtex backend. This is done by finding this in `main.tex`: *`backend=bibtex`* and changing it to *`backend=biber`*. You will then need to delete all auxiliary BibTeX files and navigate to the template directory in your terminal (command prompt). Once there, simply type `biber main` and biber will compile your bibliography. You can then compile `main.tex` as normal and your bibliography will be updated. An alternative is to set up your LaTeX editor to compile with biber instead of bibtex, see here for how to do this for various editors.

3.7.4 Tables

Tables are an important way of displaying your results, below is an example table which was generated with this code:

```
\begin{table}
\caption{The effects of treatments X and Y on the four groups studied.}
\label{tab:treatments}
\centering
\begin{tabular}{l l l}
\toprule
\tabhead{Groups} & \tabhead{Treatment X} & \tabhead{Treatment Y} \\
\midrule
1 & 0.2 & 0.8\\
2 & 0.17 & 0.7\\
3 & 0.24 & 0.75\\
4 & 0.68 & 0.3\\
\bottomrule\\
\end{tabular}
\end{table}
```

You can reference tables with `\ref{<label>}` where the label is defined within the table environment. See `Chapter1.tex` for an example of the label and citation (e.g. Table 3.1).

TABLE 3.1: The effects of treatments X and Y on the four groups studied.

Groups	Treatment X	Treatment Y
1	0.2	0.8
2	0.17	0.7
3	0.24	0.75
4	0.68	0.3

3.7.5 Figures

There will hopefully be many figures in your thesis (that should be placed in the *Figures* folder). The way to insert figures into your thesis is to use a code template like this:

```
\begin{figure}
\centering
\includegraphics{Figures/Electron}
\decoRule
\caption[An Electron]{An electron (artist's impression).}
\label{fig:Electron}
\end{figure}
```

Also look in the source file. Putting this code into the source file produces the picture of the electron that you can see in the figure below.

Sometimes figures don't always appear where you write them in the source. The placement depends on how much space there is on the page for the figure. Sometimes there is not enough room to fit a figure directly where it should go (in relation to the text) and so LaTeX puts it at the top of the next page. Positioning figures is the job of LaTeX and so you should only worry about making them look good!

Figures usually should have captions just in case you need to refer to them (such as in Figure 3.1). The `\caption` command contains two parts, the first part, inside the square brackets is the title that will appear in the *List of Figures*, and so should be short. The second part in the curly brackets should contain the longer and more descriptive caption text.

The `\decoRule` command is optional and simply puts an aesthetic horizontal line below the image. If you do this for one image, do it for all of them.

LaTeX is capable of using images in pdf, jpg and png format.

FIGURE 3.1: An electron (artist's impression).

3.7.6 Typesetting mathematics

If your thesis is going to contain heavy mathematical content, be sure that LaTeX will make it look beautiful, even though it won't be able to solve the equations for you.

The "Not So Short Introduction to LaTeX" (available on CTAN) should tell you everything you need to know for most cases of typesetting mathematics. If you need more information, a much more thorough mathematical guide is available from the AMS called, "A Short Math Guide to LaTeX" and can be downloaded from: `ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf`

There are many different LaTeX symbols to remember, luckily you can find the most common symbols in The Comprehensive LaTeX Symbol List.

You can write an equation, which is automatically given an equation number by LaTeX like this:

```
\begin{equation}
E = mc^{2}
\label{eqn:Einstein}
\end{equation}
```

This will produce Einstein's famous energy-matter equivalence equation:

$$E = mc^2 \tag{3.1}$$

All equations you write (which are not in the middle of paragraph text) are automatically given equation numbers by LaTeX. If you don't want a particular equation numbered, use the unnumbered form:

$$a^2 = 4$$

3.8 Sectioning and Subsectioning

You should break your thesis up into nice, bite-sized sections and subsections. LaTeX automatically builds a table of Contents by looking at all the `\chapter{}`, `\section{}` and `\subsection{}` commands you write in the source.

The Table of Contents should only list the sections to three (3) levels. A `chapter{}` is level zero (0). A `\section{}` is level one (1) and so a `\subsection{}` is level two (2). In your thesis it is likely that you will even use a `subsubsection{}`, which is level three (3). The depth to which the Table of Contents is formatted is set within `MastersDoctoralThesis.cls`. If you need this changed, you can do it in `main.tex`.

3.9 In Closing

You have reached the end of this mini-guide. You can now rename or overwrite this pdf file and begin writing your own `Chapter1.tex` and the rest of your thesis. The easy work of setting up the structure and framework has been taken care of for you. It's now your job to fill it out!

Good luck and have lots of fun!

Guide written by —
Sunil Patel: www.sunilpatel.co.uk
Vel: LaTeXTemplates.com

Appendix A

Frequently Asked Questions

A.1 How do I change the colors of links?

The color of links can be changed to your liking using:

`\hypersetup{urlcolor=red}`, or

`\hypersetup{citecolor=green}`, or

`\hypersetup{allcolor=blue}`.

If you want to completely hide the links, you can use:

`\hypersetup{allcolors=.}`, or even better:

`\hypersetup{hidelinks}`.

If you want to have obvious links in the PDF but not the printed text, use:

`\hypersetup{colorlinks=false}`.

Bibliography

Alvina, Jessalyn et al. (2014). "RouteLens: Easy Route Following for Map Applications". In: *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*. AVI '14. Como, Italy: ACM, pp. 125–128. ISBN: 978-1-4503-2775-6. DOI: 10.1145/2598153.2598200. URL: http://doi.acm.org/10.1145/2598153.2598200.

Arnold, A. S. et al. (Mar. 1998). "A Simple Extended-Cavity Diode Laser". In: *Review of Scientific Instruments* 69.3, pp. 1236–1239. URL: http://link.aip.org/link/?RSI/69/1236/1.

Ayachit, Utkarsh (2015). *The ParaView Guide: A Parallel Visualization Application*. USA: Kitware, Inc. ISBN: 1930934300, 9781930934306.

Bruckner, S. and M. E. Groller (2006). "Exploded Views for Volume Data". In: *IEEE Transactions on Visualization and Computer Graphics* 12.5, pp. 1077–1084. ISSN: 1077-2626. DOI: 10.1109/TVCG.2006.140.

Childs, Hank et al. (2011). "VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data". In: *Proceedings of SciDAC 2011*. Denver, CO.

Correa, Carlos, Debora Silver, and Mi Chen (Nov. 2007). "Illustrative Deformation for Data Exploration". In: *IEEE Transactions on Visualization and Computer Graphics* 13.6, pp. 1320–1327. ISSN: 1077-2626. DOI: 10.1109/TVCG.2007.70565. URL: http://dx.doi.org/10.1109/TVCG.2007.70565.

Correa, Carlos, Deborah Silver, and Min Chen (Sept. 2006). "Feature Aligned Volume Manipulation for Illustration and Visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 12.5, pp. 1069–1076. ISSN: 1077-2626. DOI: 10.1109/TVCG.2006.144. URL: http://dx.doi.org/10.1109/TVCG.2006.144.

Cui, J. et al. (2010). "A Curved Ray Camera for Handling Occlusions through Continuous Multiperspective Visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 16.6, pp. 1235–1242. ISSN: 1077-2626. DOI: 10.1109/TVCG.2010.127.

Fogal, Thomas and Jens Krüger (2010). "Tuvok, an Architecture for Large Scale Volume Rendering". In: *Proceedings of the 15th International Workshop on Vision, Modeling, and Visualization*. URL: http://www.sci.utah.edu/~tfogal/academic/tuvok/Fogal-Tuvok.pdf.

Guo, H., H. Xiao, and X. Yuan (2011). "Multi-dimensional transfer function design based on flexible dimension projection embedded in parallel coordinates". In: *2011 IEEE Pacific Visualization Symposium*, pp. 19–26. DOI: 10.1109/PACIFICVIS.2011.5742368.

Hawthorn, C. J., K. P. Weber, and R. E. Scholten (Dec. 2001). "Littrow Configuration Tunable External Cavity Diode Laser with Fixed Direction Output Beam". In: *Review of Scientific Instruments* 72.12, pp. 4477–4479. URL: http://link.aip.org/link/?RSI/72/4477/1.

Hsu, Wei-Hsien, Kwan-Liu Ma, and Carlos Correa (Dec. 2011). "A Rendering Framework for Multiscale Views of 3D Models". In: *ACM Trans. Graph.* 30.6, 131:1–

131:10. ISSN: 0730-0301. DOI: 10.1145/2070781.2024165. URL: http://doi.acm.org/10.1145/2070781.2024165.

Hurter, C. et al. (2014). "Color Tunneling: Interactive Exploration and Selection in Volumetric Datasets". In: *2014 IEEE Pacific Visualization Symposium*, pp. 225–232. DOI: 10.1109/PacificVis.2014.61.

Islam, S., D. Silver, and M. Chen (2007). "Volume Splitting and Its Applications". In: *IEEE Transactions on Visualization and Computer Graphics* 13.2, pp. 193–203. ISSN: 1077-2626. DOI: 10.1109/TVCG.2007.48.

Kincaid, Robert (Nov. 2010). "SignalLens: Focus+Context Applied to Electronic Time Series". In: *IEEE Transactions on Visualization and Computer Graphics* 16.6, pp. 900–907. ISSN: 1077-2626. DOI: 10.1109/TVCG.2010.193. URL: http://dx.doi.org/10.1109/TVCG.2010.193.

Kniss, J., G. Kindlmann, and C. Hansen (2002). "Multidimensional transfer functions for interactive volume rendering". In: *IEEE Transactions on Visualization and Computer Graphics* 8.3, pp. 270–285. ISSN: 1077-2626. DOI: 10.1109/TVCG.2002.1021579.

Li, J. et al. (2007). "Classification for Volume Rendering of Industrial CT Based on Minimum Cross Entropy". In: *2007 International Conference on Mechatronics and Automation*, pp. 2710–2715. DOI: 10.1109/ICMA.2007.4303986.

McGuffin, M. J., L. Tancau, and R. Balakrishnan (2003). "Using deformations for browsing volumetric data". In: *IEEE Visualization, 2003. VIS 2003.* Pp. 401–408. DOI: 10.1109/VISUAL.2003.1250400.

Pindat, Cyprien et al. (2012). "JellyLens: Content-aware Adaptive Lenses". In: *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. UIST '12. Cambridge, Massachusetts, USA: ACM, pp. 261–270. ISBN: 978-1-4503-1580-7. DOI: 10.1145/2380116.2380150. URL: http://doi.acm.org/10.1145/2380116.2380150.

Thiede, Conrad, Georg Fuchs, and Heidrun Schumann (2008). "Smart Lenses". In: *Smart Graphics: 9th International Symposium, SG 2008, Rennes, France, August 27-29, 2008. Proceedings*. Ed. by Andreas Butz et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 178–189. ISBN: 978-3-540-85412-8. DOI: 10.1007/978-3-540-85412-8_16. URL: http://dx.doi.org/10.1007/978-3-540-85412-8_16.

Tominski, C. et al. (2006). "Fisheye Tree Views and Lenses for Graph Visualization". In: *Tenth International Conference on Information Visualisation (IV'06)*, pp. 17–24. DOI: 10.1109/IV.2006.54.

Tominski, C. et al. (2016). "Interactive Lenses for Visualization: An Extended Survey". In: *Computer Graphics Forum*, n/a–n/a. ISSN: 1467-8659. DOI: 10.1111/cgf.12871. URL: http://dx.doi.org/10.1111/cgf.12871.

Tominski, Christian (Jan. 2011). "Event-based Concepts for User-driven Visualization". In: *Information Visualization* 10.1, pp. 65–81. ISSN: 1473-8716. DOI: 10.1057/ivs.2009.32. URL: http://dx.doi.org/10.1057/ivs.2009.32.

Tong, X., C. Li, and H. W. Shen (2017). "GlyphLens: View-Dependent Occlusion Management in the Interactive Glyph Visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 23.1, pp. 891–900. ISSN: 1077-2626. DOI: 10.1109/TVCG.2016.2599049.

Tong, X. et al. (2016). "View-Dependent Streamline Deformation and Exploration". In: *IEEE Transactions on Visualization and Computer Graphics* 22.7, pp. 1788–1801. ISSN: 1077-2626. DOI: 10.1109/TVCG.2015.2502583.

Wang, L. et al. (2005). "The magic volume lens: an interactive focus+context technique for volume rendering". In: *VIS 05. IEEE Visualization, 2005.* Pp. 367–374. DOI: 10.1109/VISUAL.2005.1532818.

Wang, Y. et al. (2011). "Efficient Volume Exploration Using the Gaussian Mixture Model". In: *IEEE Transactions on Visualization and Computer Graphics* 17.11, pp. 1560–1573. ISSN: 1077-2626. DOI: 10.1109/TVCG.2011.97.

Wieman, Carl E. and Leo Hollberg (Jan. 1991). "Using Diode Lasers for Atomic Physics". In: *Review of Scientific Instruments* 62.1, pp. 1–20. URL: http://link.aip.org/link/?RSI/62/1/1.

Wu, M. L. and V. Popescu (2016). "Multiperspective Focus + Context Visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 22.5, pp. 1555–1567. ISSN: 1077-2626. DOI: 10.1109/TVCG.2015.2443804.