UNIVERSITY NAME

DOCTORAL THESIS

Interactive visualization for volumetric datasets

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

July 20, 2018

Declaration of Authorship

I, John SMITH, declare that this thesis titled, "Interactive visualization for volumetric datasets " 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

Interactive visualization for volumetric datasets

by John SMITH

abstract here

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

1.1 Context

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.

Volumetric data is very common nowadays. The importance of this dataset type will grow rapidly due to the development of the 3D data acquisition field, and the possibilities to perform an advanced visualization on a modern office workstation with an interactive framerate.

The dataset can be captured by various technologies, e.g. **MRI (Magnetic resonance imaging)** , **CT (computed tomography)** , **PET (Positron-emission tomography)** , **USCT (Ultrasound computer tomography) echolocation** . It also can be produced by physical simulations, for example, fluid dynamics or particle systems. The set of technologies mentioned before demonstrates that volumetric information plays an important role in medicine. It is used for an advanced cancer detection, visualization of aneurysms, and treatment planning. This kind of data is also very useful for non-destructive material testing via computer tomography or ultrasound. In addition, huge three-dimensional dataset is produced by geoseismic research.

An **MRI (Magnetic resonance imaging)** is a radiology technique scan that uses magnetism, radio waves, and a computer to produce images of body structures. The MRI scanner is a tube surrounded by a giant circular magnet. The patient is placed

on a movable bed that is inserted into the magnet. The magnet creates a strong magnetic field that aligns the protons of hydrogen atoms, which are then exposed to a beam of radio waves. This spins the various protons of the body, and they produce a faint signal that is detected by the receiver portion of the MRI scanner. A computer processes the receiver information, which produces an image. MRI image and resolution is quite detailed, and it can detect tiny changes of structures within the body. For some procedures, contrast agents, such as gadolinium, are used to increase the accuracy of the images.
An MRI scan can be used as an extremely accurate method of disease detection throughout the body and is most often used after the other testing fails to provide sufficient information to confirm a patient's diagnosis. In the head, trauma to the brain can be seen as bleeding or swelling. Other abnormalities often found include brain aneurysms, stroke, tumors of the brain, as well as tumors or inflammation of the spine.
Neurosurgeons use an MRI scan not only in defining brain anatomy, but also in evaluating the integrity of the spinal cord after trauma. It is also used when considering problems associated with the vertebrae or inter-vertebral discs of the spine. An MRI scan can evaluate the structure of the heart and aorta, where it can detect aneurysms or tears. MRI scans are not the first line of imaging test for these issues or in cases of trauma.
It provides valuable information on glands and organs within the abdomen, and accurate information about the structure of the joints, soft tissues, and bones of the body. Often, surgery can be deferred or more accurately directed after knowing the results of an MRI scan.

Computed tomography (CT) is a diagnostic imaging test used to create detailed images of internal organs, bones, soft tissue and blood vessels. The cross-sectional images generated during a CT scan can be reformatted in multiple planes, and can even generate three-dimensional images which can be viewed on a computer monitor, printed on film or transferred to electronic media. CT scanning is often the best method for detecting many different cancers since the images allow to confirm the presence of a tumor and determine its size and location. CT is fast, painless, noninvasive and accurate. In emergency cases, it can reveal internal injuries and bleeding quickly enough to help save lives.
Computed tomography (CT) of the body uses sophisticated x-ray technology to help detect a variety of diseases and conditions. CT scanning is fast, painless, noninvasive and accurate. Industrial CT Scanning is a process which utilizes X-ray equipment to produce 3D representations of components both externally and internally. Industrial CT scanning has been utilized in many areas of industry for internal inspection of components. Some of the key uses for CT scanning have been flaw detection, failure analysis, meteorology, assembly analysis, image-based finite element methods and reverse engineering applications. CT scanning is also employed in the imaging and conservation of museum artifacts.

PET (Positron-emission tomography) uses small amounts of radioactive materials called radio-tracers, a special camera and a computer to help evaluate your organ and tissue functions. By identifying body changes at the cellular level, PET may detect the early onset of disease before it is evident on other imaging tests. It is a type of nuclear medicine imaging. Nuclear medicine is a branch of medical imaging that uses small amounts of radioactive material to diagnose and determine

the severity of or treat a variety of diseases, including many types of cancers, heart disease, gastrointestinal, endocrine, neurological disorders and other abnormalities within the body. Because nuclear medicine procedures are able to pinpoint molecular activity within the body, they offer the potential to identify disease in its earliest stages as well as a patient's immediate response to therapeutic interventions. Nuclear medicine images can be superimposed with computed tomography (CT) or magnetic resonance imaging (MRI) to produce special views, a practice known as image fusion or co-registration. These views allow the information from two different exams to be correlated and interpreted on one image, leading to more precise information and accurate diagnoses.

USCT (Ultrasound computer tomography) echolocation uses ultrasound waves for creating images. In the first measurement step a defined ultrasound wave is generated with ultrasound transducers, transmitted in direction of the measurement object and received with other or the same ultrasound transducers. While traversing and interacting with the object the ultrasound wave is changed by the object and carries now information about the object. After being recorded the information from the modulated waves can be extracted and used to create an image of the object in a second step. Unlike X-ray or other physical properties which provide typically only one information, ultrasound provides multiple information of the object for imaging: the attenuation the wave's sound pressure experiences indicate on the object's attenuation coefficient, the time-of-flight of the wave gives the speed of sound information, and the scattered wave indicates on the echogenicity of the object (e.g. refraction index, surface morphology, etc.)

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.

During this thesis, we mainly focused on two main topics:

- A user study where we investigated the specific activity of baggage inspection, and proposed an interactive visualization system to support their volumetric data exploration according to the requirements and constraints of this field.
- Focus+Context techniques for occlusion management in volumetric data visualization.

1.2 Problem

Interacting with volumetric 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.).

1.2.1 Baggage inspection

Since volumetric data-sets are more and more used in many areas thanks to technological breakthroughs, switching from the old systems working with 2D images to the newest ones with 3D is not straightforward and easy. In the field of baggage inspection, the displayed 2D scanned image can suffer from four issues or dissimulation strategy.

Superposition: A threat (e.g. prohibited object like knife, cutter...) may be sheltered behind dense materials. Sometimes, it is possible to see through these blind shield using some functionalities such as high penetration (enhanced X-ray power) or image processing (contrast improvement).

Location: Depending on its location inside the luggage, a threat can be difficult to detect. Objects located in the corners, in the edges or inside the luggage's frame are very difficult to identify.

Dissociation: Another way to dissimulate a threat is to separate and to spread parts of it in the luggage (weapon or explosive are composed of many separated items like the trigger, the cannon...). This dissociation can be combined with other dissimulation techniques.

Lure: An ill-intentioned individual may use a lure to hide the real threat. For instance, a minor threat like a small scissors may be clearly visible and catch security agent's attention while a more important threat remains hidden.

3D baggage scan exploration are one potential solution of such limitations. Few systems investigated this activity domain with interactive volumetric exploration tools like Li et al., 2012. Even if extensive works have been done in medical 3D scan exploration and manipulation by Preim and Botha, 2013, there is a great opportunity

to adapt and develop new interaction and data manipulation techniques to support 3D baggage exploration.

1.2.2 Occlusion management using Focus+Context techniques

Direct volume rendering (DVR) is a pervasive visualization technique for displaying 3D scalar fields with applications in engineering, material sciences, and medical imaging sciences. However widely adopted, and able to handle large datasets at interactive rates, DVR inherently suffers from the problem of *occlusion*: Structures of interest located deep in the volume, called next *targets*, can be hard to spot and/or explore.

To address this issue, various techniques have been designed including transfer functions, segmentation, selection, and clipping. Yet, all such techniques have limitations. *Global* mechanisms, like transfer function editing, can remove both occluders and targets if these have similar densities. In certain applications, carefully designed transfer functions exist and should be used without (significant) modifications to facilitate understanding and user training like with Wu and Qu, 2007. *Local* mechanisms like segmentation, selection, or clipping are more effective in manipulating data confined to a given spatial region. Yet, many such mechanisms assume that one can easily and accurately select targets to remove them (occluders) or keep them (occluded). This is hard to do when *e.g.* one does not have direct access to the targets, or when significant 3D interaction is required to select occluder(s).

A different way to handle occlusion is to use *lenses*. These are flexible lightweight tools which enable local and temporary modifications of the DVR to reveal targets while keeping the global visualization context Carpendale, Cowperthwaite, and Fracchia, 1997; Tominski et al., 2016; Tominski et al., 2012. However, efficiently selecting the target and removing all in-between occluders is still challenging. More specifically, most existing occlusion management techniques do not simultaneously meet all following requirements:

- rapidly create an unobstructed view of the target (R1),
- allow a flexible local exploration of the target zone (R2),
- keep the context in which the target is visually embedded (R3),
- handle data-sets where the target and occluders cannot be separated by transfer function manipulations (R4).

1.3 Research Question

The two problems presented in the previous section (section 1.2) have something in common. In fact both issues are related to the management of occlusion and the analysis of the context.

First, with the baggage inspection, being able to display correctly the volume and provide interactive tool to explore while keeping in mind the whole context is really important.

Second, although existing occlusion management are effective, most of them fail to keep the context while exploring a local region of interest. In, addition they do not not provide enough flexibility to interact with the region of interest while the rest of the volume remains intact.

From these observations, this thesis addresses the following research question:

Can we create interactive techniques to explore volumetric data-sets while taking into account the context , and providing flexibility to the user ?

1.4 Thesis outline

This thesis present two main contributions. The first one is a user Study of the interactive exploration of 3D scanned baggage.

As second contribution, we increase the flexibility of lenses for direct volume rendering exploration to jointly cover all above requirements in subsection 1.2.2. We propose a focus-and-context (F+C) lens that combines a distortion technique, which pushes aside the occluding objects, with a fish-eye field of view, to provide a better perspective on targets.

Before detailing those contributions, this manuscript starts with a presentation of existing work in volume rendering algorithms, occlusion management strategies, and volume rendering in mobile devices(smartphones, Virtual Reality, and Augmented Reality). First we present the existing algorithms to render volumetric data-sets, then we discuss the existing strategies of occlusion management. finally we describe the previous work on volume rendering in mobile devices.

The third chapter present the user study of the interactive exploration of 3D scanned baggage. First, we present the activity of airport security agents and the difficulties they face in order to highlight the main requirements for the new 3D systems. Second, we propose a set of GPGPU interactions to explore the baggage. Then we illustrate the efficiency of the proposed interactions with two use cases or scenarios. In addition, we highlight the technical constraints and the implementation of these interaction techniques. Finally we conclude this chapter and discuss the pertinence of the work presented in this chapter.

The fourth chapter proposes a focus-and-context (F+C) lens that combines a distortion technique, which pushes aside the occluding objects, with a fish-eye field of view, to provide a better perspective on targets. First we exhaustively define the requirements already presented in section 1.2. Second, we present the principle of our novel lens by describing all the parameters and their influences. Third, we show how this lens has been implemented on the GPU. Furthermore, we illustrate this

novel lens thanks to five different scenarios from different fields of activity. Finally we discuss the advantages and the shortcomings of this lens, then conclude this chapter.

The fifth chapter deals with volume rendering on mobile devices (Virtual Reality, Augmented Reality, Mixed Reality). In fact, This chapter presents an early on going work on this topic. First we define role of the stereoscopic 3D technique. Second, we present address the case of virtual reality (VR). Third, we deal with Augmented reality (AR). Then, we study volume rendering in mixed reality (MR). Finally we conclude and discuss the perspective of this work.

Finally, we present the conclusion of the thesis and directions for future work.

Chapter 2

State of the art

2.1 Introduction

In this Chapter, we carry out a presentation of existing work in volume rendering algorithms, occlusion management strategies, and volume rendering in mobile devices(Mixed Reality, Virtual Reality, and Augmented Reality). First we present the existing algorithms to render volumetric data-sets that can be categorized into two major types: Direct Volume rendering (DVR), and Surface-fitting algorithms. Then we discuss the existing strategies of occlusion management. Actually, we talk about the transfer function, the segmentation, an deformations as well as Focus+context techniques. Finally we describe the previous work on volume rendering in mobile devices. More precisely, we deal with Mixed Reality, Virtual Reality, and Augmented Reality.

2.2 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

2.2.1 Direct Volume Rendering

2.2.1.1 Texture mapping

2.2.1.1.1 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.

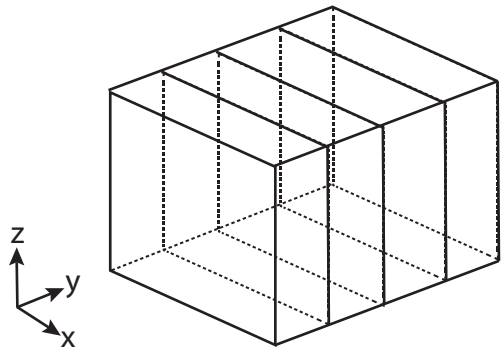


FIGURE 2.1: 2D slices of the volume.

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.

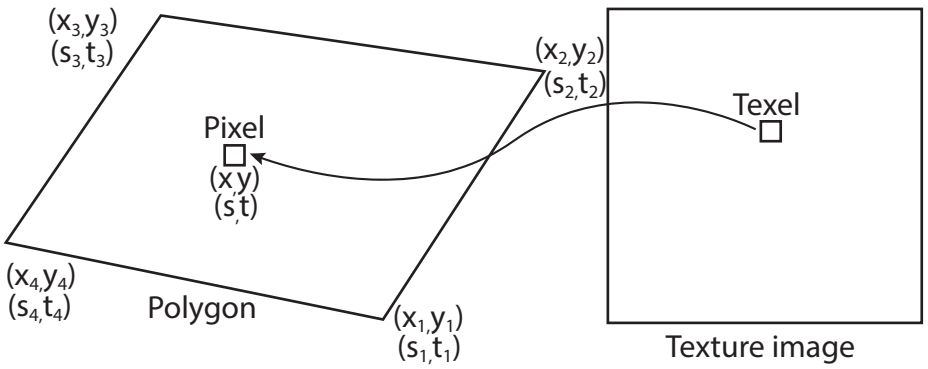


FIGURE 2.2: Calculating color and opacity of a pixel inside a texture-mapped polygon.

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.

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**

2.2.1.1.2 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.

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.

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.

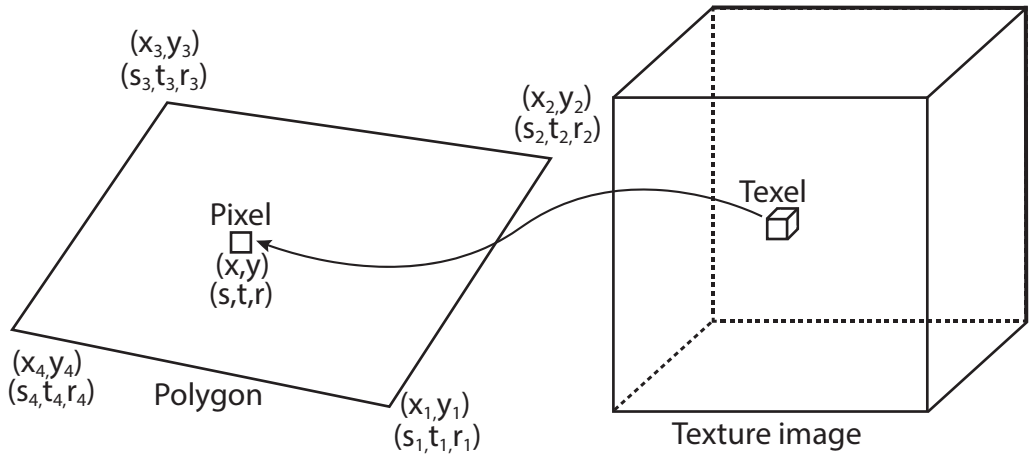


FIGURE 2.3: Calculating color and opacity of a pixel inside a texture-mapped polygon using 3D texture.

2.2.1.2 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 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

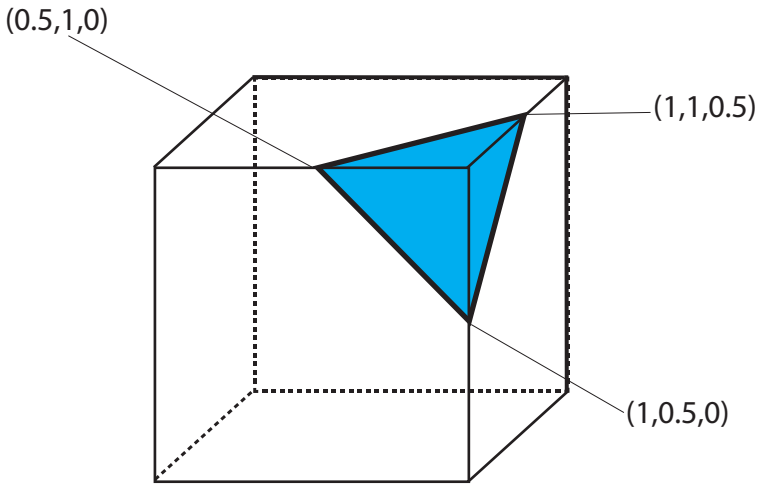


FIGURE 2.4: Calculating vertex coordinates and texture coordinates for an oblique slice. In this figure, the vertex and texture coordinates are identical.

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).

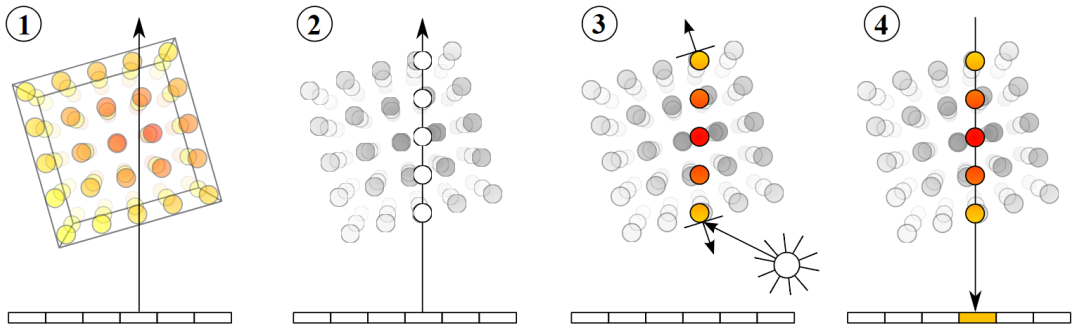


FIGURE 2.5: The four steps of the volume raycasting algorithm

2.2.2 Surface-fitting

This subsection describes an algorithm for creating a polygonal surface representation of an isosurface of a 3D scalar field. A common name for this type of problem is the so called **"marching cubes"** algorithm. It combines simplicity with high speed since it works almost entirely on lookup tables.

The fundamental problem is to form a facet approximation to an isosurface through a scalar field sampled on a rectangular 3D grid. Given one grid cell defined by its vertices and scalar values at each vertex, it is necessary to create planar facets that best represent the isosurface through that grid cell. The isosurface may not be pass through the grid cell, it may cut off any one of the vertices, or it may pass through in any one of a number of more complicated ways. Each possibility will be characterised by the number of vertices that have values above or below the isosurface. If one vertex is above the isosurface say and an adjacent vertex is below the isosurface then we know the isosurface cuts the edge between these two vertices. The position that it cuts the edge will be linearly interpolated, the ratio of the length between the two vertices will be the same as the ratio of the isosurface value to the values at the vertices of the grid cell.

There are two major components of the Marching cubes algorithm proposed by Lorensen and Cline, 1987. The first is deciding how to define the section or sections of surface which chop up an individual cube. If we classify each corner as either being below or above the isovalue, there are 256 possible configurations of corner classifications. Two of these are trivial; where all points are inside or outside the cube does not contribute to the isosurface. For all other configurations we need to determine where, along each cube edge, the isosurface crosses, and use these edge intersection points to create one or more triangular patches for the isosurface.

If you account for symmetries, there are actually only 14 unique configurations in the remaining 254 possibilities. When there is only one corner less than the isovalue, this forms a single triangle which intersects the edges which meet at this corner, with the patch normal facing away from the corner. Obviously there are 8 related configurations of this sort . By reversing the normal we get 8 configurations which have 7 corners less than the isovalue. We don't consider these really unique, however. For configurations with 2 corners less than the isovalue, there are 3 unique configurations, depending on whether the corners belong to the same edge, belong the same face of the cube, or are diagonally positioned relative to each other. For configurations with 3 corners less than the isovalue there are again 3 unique configurations (e.g. for configuration 14), depending on whether there are 0, 1, or 2 shared edges (2 shared edges gives you an 'L' shape). There are 7 unique configurations when you have 4 corners less than the isovalue, depending on whether there are 0, 2, 3 (3 variants on this one), or 4 shared edges (e.g. for configuration 30 - again you may need to tweak the colours to see the triangle for the isolated (far) inside sphere/pixel).

Each of the non-trivial configurations results in between 1 and 4 triangles being added to the isosurface. The actual vertices themselves can be computed by interpolation along edges, or, default their location to the middle of the edge. The interpolated locations will obviously give you better shading calculations and smoother surfaces.

The quality of the final the polygonal surface representation heavily depends on the number on isosurface generated. Accoording to the implementation used and the optimization strategies we can find a better trade-off between the quality and the frame rate (see Figure 2.6).

Using isosurfaces allow to be faster than direct volume rendering. However, we

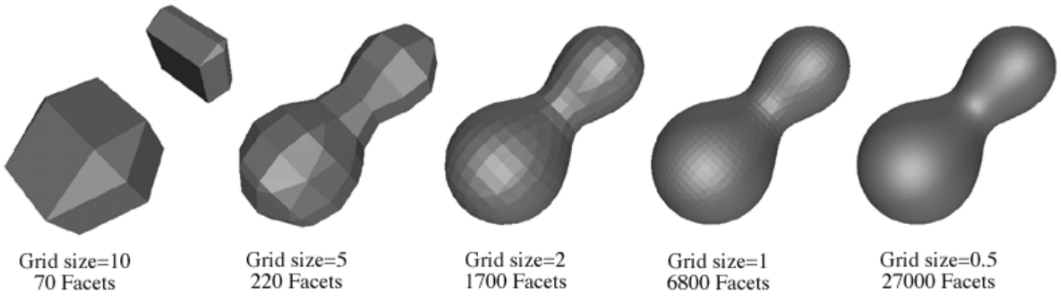


FIGURE 2.6: Two molecules, generated at different grid sizes

lose the flexibility and the precision of the DVR algorithms because of the different approximations made while extracting these isosurfaces according to a maximum number of surfaces.

2.3 Occlusion management strategies

2.3.1 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.

2.3.2 Segmentation

Many segmentation algorithms allows to extract features form images (2D) and 3D volumes. In this subsection, we present two common algorithms: **Thresholding**, and **Clustering**

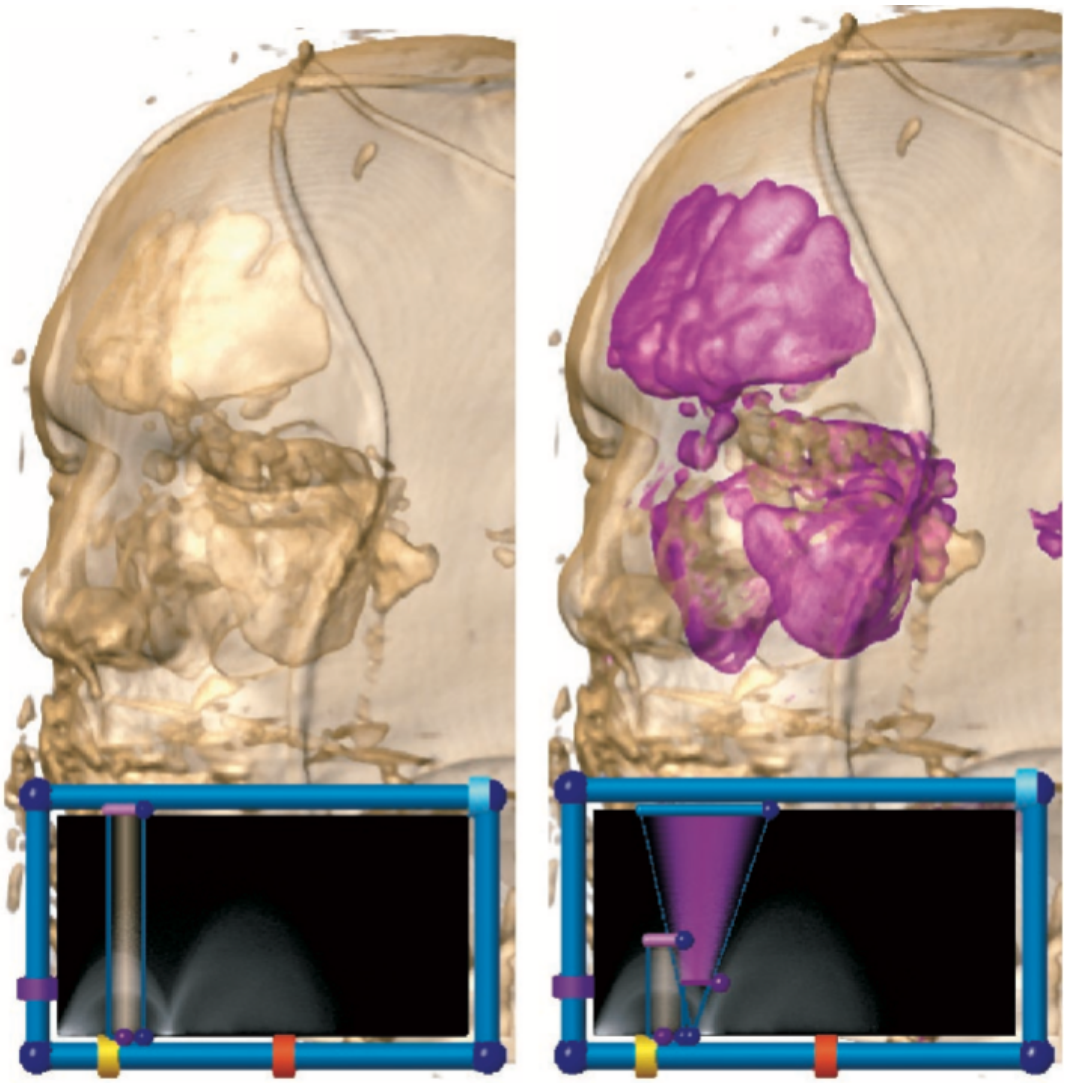


FIGURE 2.7: The two-dimensional TF editor widgets allow the user to more precisely control the classification power of the 2D TF. The triangular widget can also be skewed to better fit the desired classification domain (Kniss, Kindlmann, and Hansen, 2002)

2.3.2.1 Thresholding

Scalar images can be segmented using thresholding approaches by partitioning the volume intensities. This methodology attempts to determine an intensity value that can separate the signal into a desired number of classes. Segmented volumes can be achieved by clustering all pixels with intensities larger than the threshold value into one class, and all others into another. In many applications, the threshold values selection can be done depending on the basis of histogram. Multithresholding occurs when more than one threshold value is determined (see Sahoo et al., 1988). The voxels of a certain object are not necessarily connected after thresholding because this technique does not consider the spatial characteristics of an image, thus causing it to be sensitive to noise and intensity fluctuations. For this reason it cannot be easily applied to many medical imaging modalities. These drawbacks essentially corrupt the

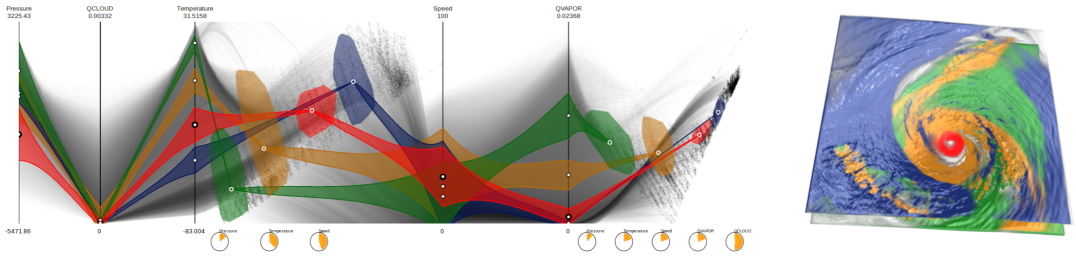


FIGURE 2.8: Parallel coordinates with embedded multidimensional scale plots for editing multidimensional TFs(Guo, Xiao, and Yuan, 2011)

histogram of the image-making partitioning via the selection of more problematic appropriate thresholds.

3D thresholding method is similar to the 2D approaches where thresholding process is applied on all pixels in the volume instead of that in the plane.

2.3.2.2 Clustering

Clustering technique is the process of classifying each group of pixels in an image into one class, each class has the same or similar properties which evaluate a specific part of an image or a volume. Each class is highlighted in the segmented volume to illustrate the volume as a number of separated regions, and the region of interest may be one of those regions. Clustering technique is based on multithreshold values which can be set depending on the image histogram. Amira et al., 2008 propose a method for segmenting medical volumes; it is the most commonly used clustering technique. -means will be used in the fourth section of this research paper as a previously available work to validate the proposed techniques in the comparison tables.

K-means clustering classifies n voxels into K clusters or classes (K less than n). This algorithm chooses the number of clusters (K) then randomly generates K clusters and determine the cluster centers. The next step is assigning each point in the volume to the nearest cluster center and finally recompute the new cluster centers. The two previous steps are repeated until the minimum variance criterion is achieved. This approach is similar to the expectation-maximization algorithm for Gaussian mixture in which they both attempt to find the centers of clusters in the volume.

2.3.3 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.
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 2.9.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.

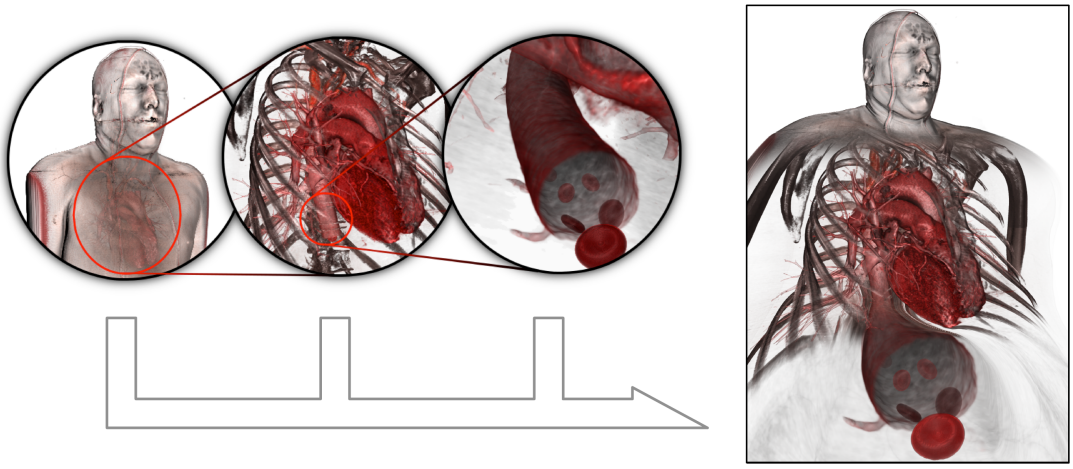


FIGURE 2.9: A Rendering Framework for Multiscale Views of 3D Models. A continuous multiscale view (right) of a volumetric human body dataset shows three different levels of detail (left three) in a single image. The image on the right is directly rendered with our multiscale framework.

Bruckner and Groller Bruckner and Groller, 2006 proposed exploded view for volume data by partitioning the volume into several segments (figure 2.10). 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:

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

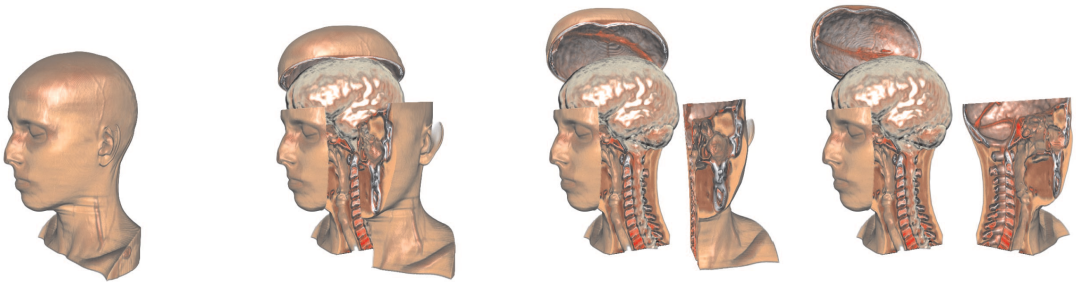


FIGURE 2.10: Interactive exploded-view illustration of a human head with increasing degrees-of-explosion. Two hinge joints are used to constrain part movement

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.

Kruger, Schneider, and Westermann, 2006 proposed ClearView, an interactive and intuitive volume visualization tool that provides the user with a simple exploration metaphor, see figure 2.11 . Following traditional technical illustrations, several 3D layers of volumetric data sets are extracted using texture-based raycasting. They are composed to produce high-quality images at high frame rates. The user guides the exploration process by moving the hotspot, a lens-like yet distortion free region, in which additional layers of the data set are augmented to convey relevant features. The proposed GPU system exploits feature-based techniques to improve the understanding of complex 3D data sets, and it utilizes image-based deferred shading to maximize performance. ClearView does not require any additional feature volumes, making the method suitable for the interactive rendering of high-resolution data sets on recent GPUs. Several shaders to intuitively convey material and shape properties are integrated into the system. To keep the user interface slim, only few parameters abstract from the technical realization. The user

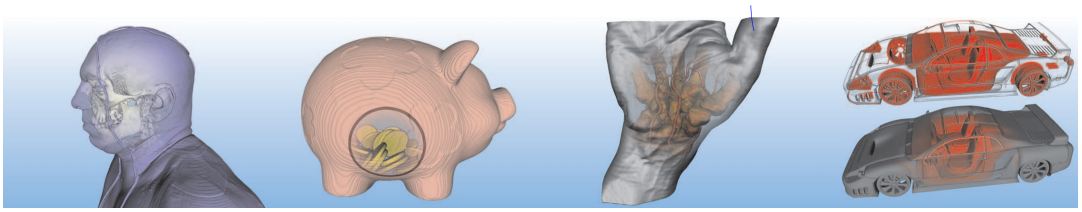


FIGURE 2.11: The ClearView system can be used to visually explore complex data sets at interactive frame rates by using a focus+context metaphor. Even the leftmost image of the 5123 visible human data set with multiple iso-surfaces was generated at about 15 fps on a 800x600 viewport.

simply positions the hotspot on the data set and selects an amount of transparency to continuously blend between focus and context information.

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.

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 2.12. 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

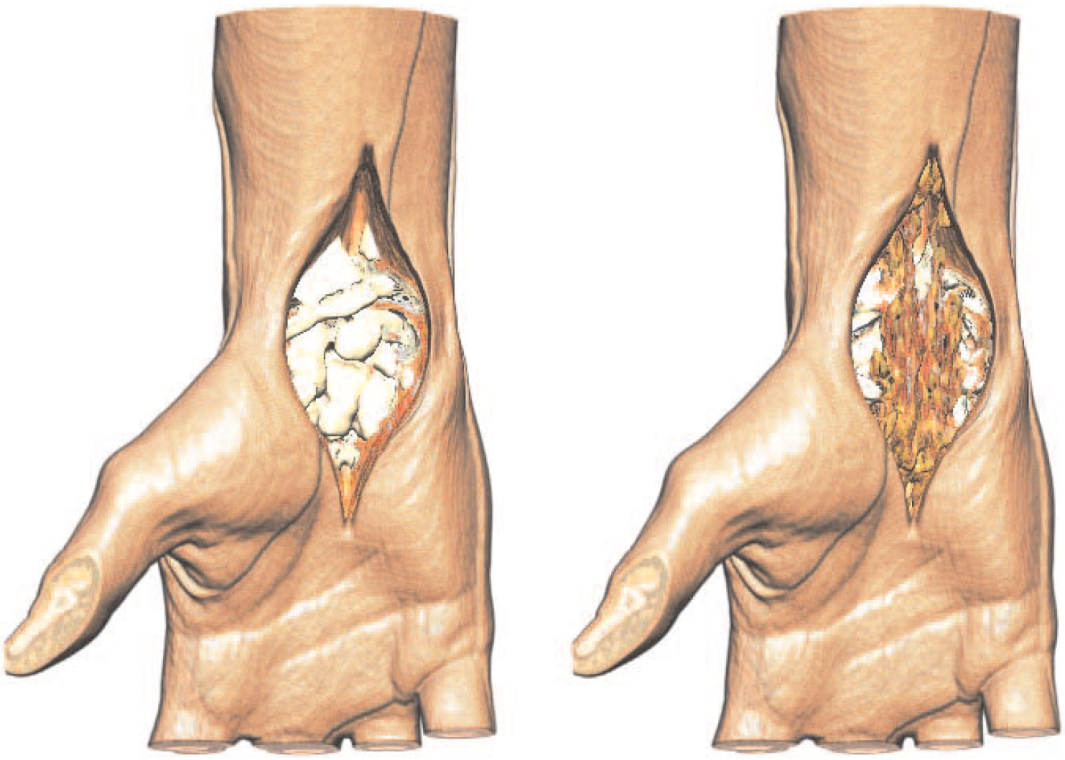
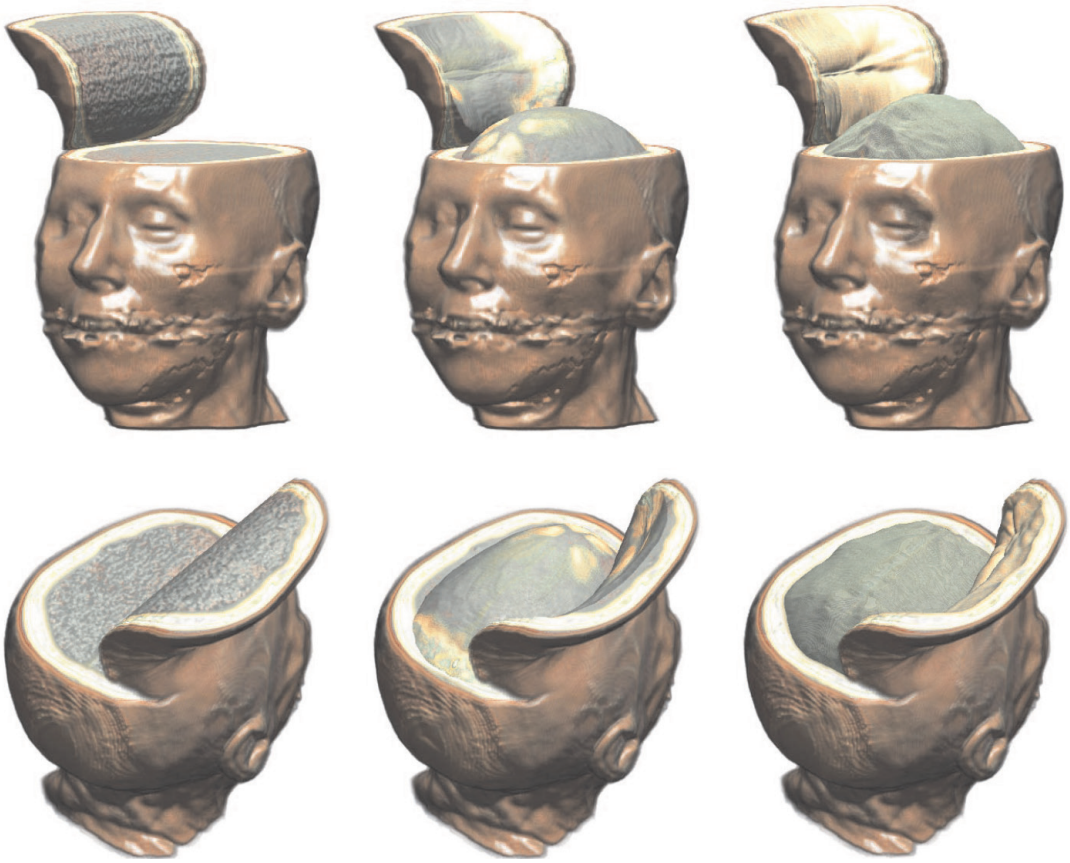


FIGURE 2.12: A feature-aligned retraction applied to a human hand data set, showing bones (left) and vessels (right)

pliers, this system allows the specification of manipulation through a collection of procedurally-defined manipulation operators.

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 2.13. 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.



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

FIGURE 2.13: 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.

- 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., 2014a 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 2.14):

- 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

Li et al., 2012 present a system for luggage visualization where any object is clearly distinguishable from its neighbors. It supports virtual unpacking by visually moving any object away from its original pose. To achieve these, we first apply a volume segmentation guided by a confidence measure that recursively splits connected regions until semantically meaningful objects are obtained, and a label volume whose voxels specifying the object IDs is generated.

First, a volumetric representation of a luggage is segmented into meaningful real-world objects. Then the segmented objects are assigned distinguishable colors depending on the viewing parameters. The original luggage volume, the segmentation

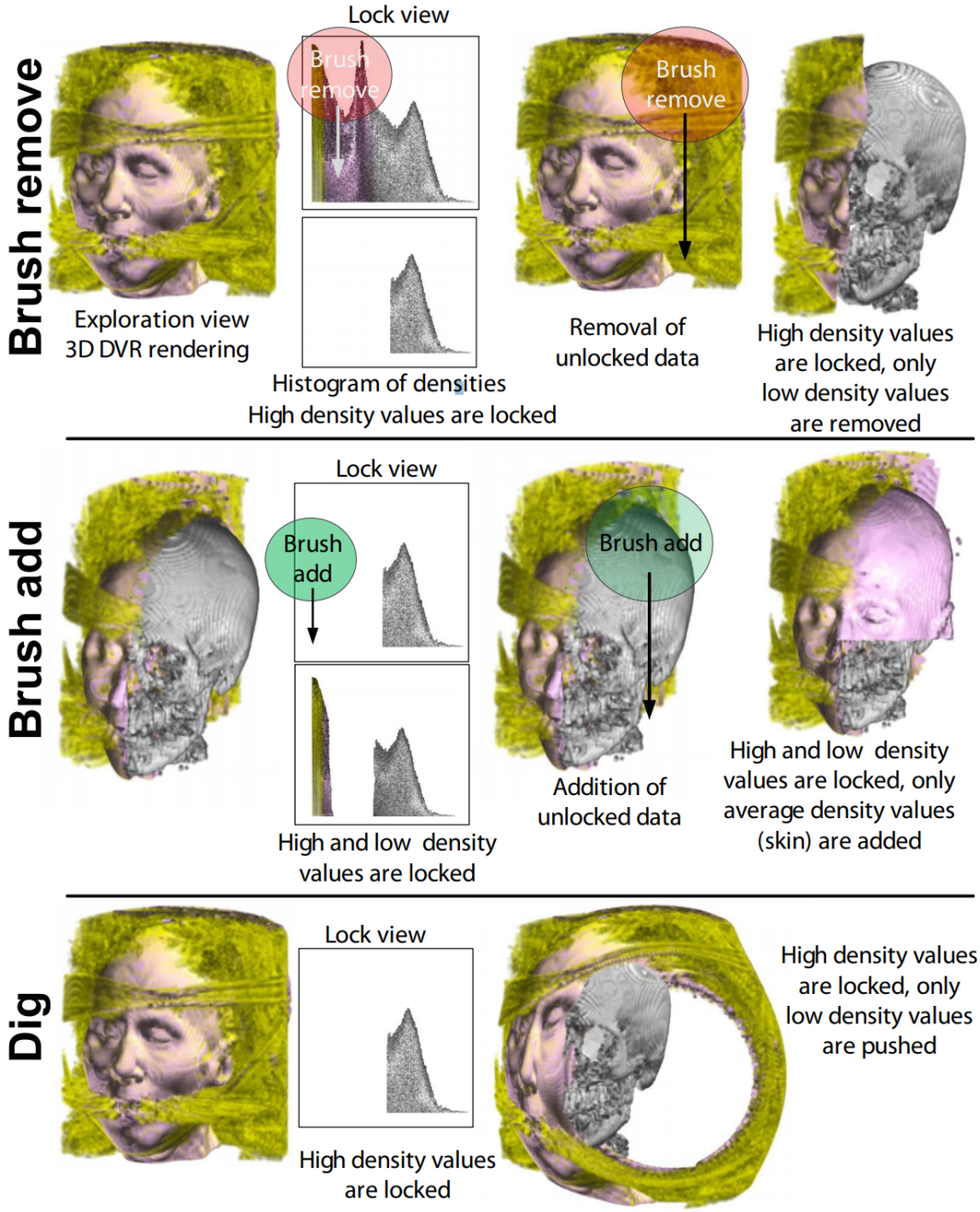


FIGURE 2.14: Brush, dig, and lock tools. Locked items are not affected by brush and dig. We used transfer functions to make air voxels transparent and standard gradient shading. We see that the head is surrounded by a large amount of uninteresting noise (yellow).

results, and the automatically assigned colors are fed into a volume rendering system. For any user defined unpacking direction, unpackable objects are determined and unpacking is visualized by showing that the transition of the objects flying from its original position to a unpacking destination. Unpacked objects can be manipulated individually, such as rotating, changing transfer function, etc, for inspection. The system keeps packed luggage, already unpacked objects, and the objects in transition in the same scene and allows repacking.

For the luggage segmentation strategy, this system utilizes a the graph partitioning approach by Grady and Schwartz, 2006 enhanced by a confidence measure of the quality of a segmentation. The confidence measure gives high scores to semantic meaningful realworld objects that are homogeneous and have well-defined boundaries against the surrounding segments, and it guides the segmentation algorithm to recursively split connected objects in an input volume until the segmentation quality cannot be improved any more.

In order to assign distinct colors to objects close to each other in an image, their system first generates a layered image of object IDs, from which whether any pair of objects are interfering each other is determined. A modified graph coloring algorithm is then computed to ensure any interfering pair are assigned distinguishable hues while the hue space is fully exploited. In order to reduce volume rendering cost as well as to keep smooth transitions across touching objects, this system adopts a tinting approach, in which the assigned hues of an object is modulated onto a master transfer function, see Figure 2.15 .

2.4 Volume rendering in mobile devices(Mixed Reality, Virtual Reality, and Augmented Reality)

The computational power of mobile devices is much stronger, and some traditional tasks on computer can be handled by mobile devices. One of the most obvious areas is graphics on mobile devices. Usually people use vertex data to represent polygons and combine multiple polygons to describe the surface of an object. The surface of an object can be reconstructed easily with polygons, but the inner structures of the object is invisible. Most of the time, such as in 3D computer games, people do not concern the structures inside the object. However, some fields including science, engineering, and medicine, visualizing inner structures of a volume data is important. Therefore, direct volume rendering (DVR) was developed to visualize volume data. As mobile devices are getting ubiquitous, people need portable equipment solutions for DVR. Although the computation time of DVR is more expensive than that of polygon data, there is still a need to implement DVR on latest mobile devices with new graphics technologies.

Xin and Wong, 2016 proposed a framework based on OpenGL ES and GLSL shaders. This mobile framework has three components, namely, rendering, histogram mapping, and transfer function design. They implemented a one-pass shader in order to allow the system work in real-time. High dimension transfer functions can be easily used in our framework without losing the user interaction intuition.

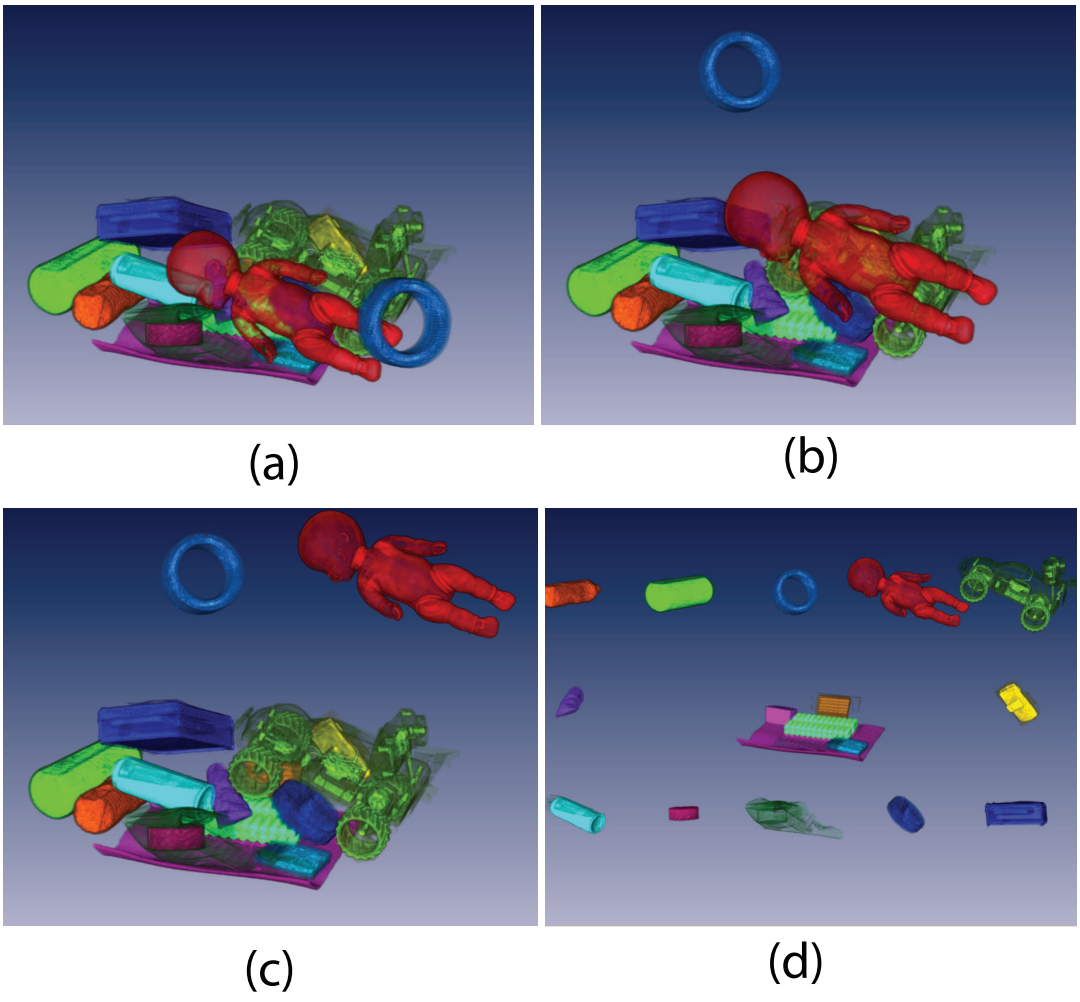


FIGURE 2.15: Virtual unpacking: (a) all objects in packed positions, (b)-(c)images extracted from the animation of unpacking the red doll, which first moves towards the camera, then it flies to the top-right corner of the image; and (e) most objects are in unpacked positions

The user interface and transfer function design steps are user friendly to the non-professional user. They proposed different interactions such as:

- **Draw touch mode check box**: he default drag action on screen is to rotate the volume object. When this check box is checked, the system will switch to draw mode. The touch or drag action will apply color onto the corresponding position on the transfer function texture.
- **2D transfer function check box**: If this check box is checked, the system will enable 2D transfer functions.
- **Compositing mode radio group**: : There are four compositing modes: X-ray, MIP, iso-surface, and full rendering.
- **Showing texture radio group**: There are four textures which can be shown: back face, final, intensity, and gradient textures.
- **Showing texture radio group**: There are four colors that can be selected in this control panel. One may design a better color selector for specific the brush color.
- **Other controls**: More controls that have not yet been implemented in this control panel. For example, the ray casting steps can be controlled, more steps bring higher render quality, fewer steps bring better performance. The pen size and sharp can also be controlled, even the 2D transfer function image can be paint directly. Due to the complex of the system, we have not yet implemented all the controls.

The rendering performance of this framework in the Xray, MIP, and iso-surface modes are around 11 FPS, almost the same. For the full rendering mode, due to its heavy computation, the rendering performance is much slower as 2 FPS. Basically we can achieve real-time DVR on this mobile device by reducing unnecessary frames.

Ard et al., 2017 proposed Neuro Imaging in Virtual Reality (NIVR), which is a visualization suite that employs various immersive visualizations to represent neuroimaging information in VR. Some established techniques, such as raymarching volume visualization, are paired with newer techniques, such as near-field rendering, to provide a broad basis of how we can leverage VR to improve visualization and navigation of neuroimaging data. Several of the neuroscientific visualization approaches presented are, to our knowledge, the first of their kind. NIVR offers not only an exploration of neuroscientific data visualization 2.16, but also a tool to expose and educate the public regarding recent advancements in the field of neuroimaging. By providing an engaging experience to explore new techniques and discoveries in neuroimaging, we hope to spark scientific interest through a broad audience.

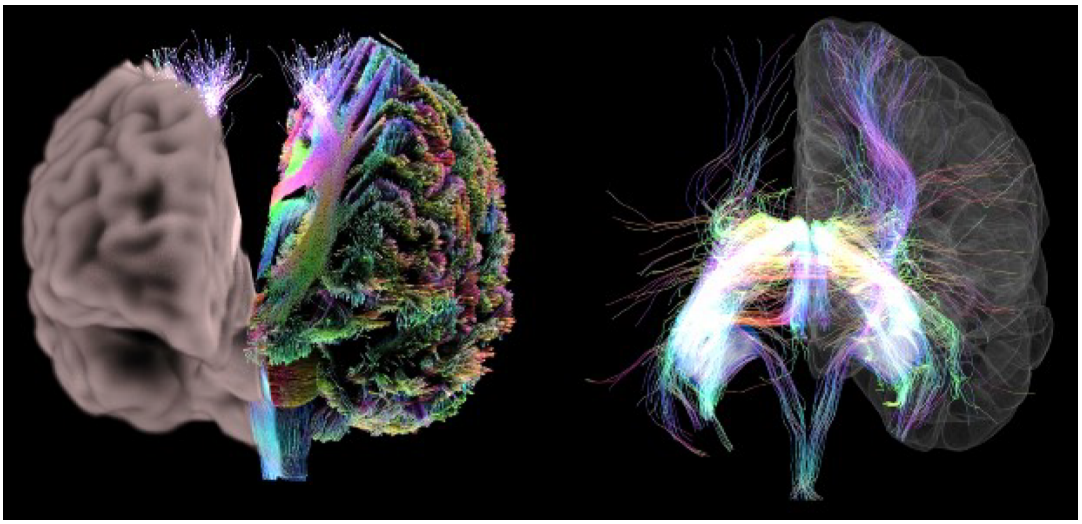


FIGURE 2.16: Near Field renderings in NIVR

Ran and Dingliana, 2016 presented a framework for rendering 3D volumetric data on augmented reality displays. The system exploits depth sensors that are becoming available in most augmented reality HMDs and mobile devices. They provided adaptations to their framework to accommodate devices with relatively low-end computational capabilities, as is common in mobile and HMD platforms designed to be highly portable. The framework achieves real-time performance across a number of platforms and provides improvements that preserve the sense of interior volumes being embedded inside real-world objects.

To virtually unpack a luggage, instead of physically partitioning the original volume, the system performs volume rendering in multiple passes and each pass is associated with the proper transfer function and geometric transformation. A brutal-force design requires the number of rendering passes equals to that of the objects in the worst case. This obviously is too expensive considering a typical luggage contains several dozens of objects. To efficiently handle a scene mixed with a volume of packed objects, unpacked objects, and the animation of objects being unpacked or restored, they have designed a layered approach. Each layer is responsible for rendering objects in a particular state as well as caches and displays the previous rendered images to avoid unnecessary rendering. The layer of unpacked objects is further decomposed into a set of 3D sprites to minimize memory usage

Chapter 3

User Study: Interactive exploration of 3D scanned baggage

3.1 Introduction

Volumetric data found in many fields, such as engineering, material sciences, medical imaging, astrophysics. Their exploration is not trivial, and heavily impacted by the users' needs. In most airports, security agents deal with such data exploration during baggage inspections. They use two types of fluoroscopic systems: X-ray and tomography scanning systems. X-ray system provides a flattened 2D baggage scan while tomography scanning system produces transversal scans, also called slices. Thanks to data processing (e.g. Deans, 2007), they can produce a full 3D scan (set of voxel with their corresponding density). Our input data is a uniformly sampled multivariate field $F : D \longrightarrow V$, $D \subset \mathbb{R}^n$,$V \subset \mathbb{R}^m$ with n = 3 (volume) and m=1 (scalar field).

3.2 Activity analysis

During this thesis, I had the opportunity to intend to training courses for security agents' instructors, and to visit an airport (Toulouse-Blagnac Airport). These helped me to study the activity of security agents and get the users' needs. The training courses lasted 3 weeks. In fact, I studied weapons and Explosives, scan analysis methods, dissimulation techniques, and fluoroscopic systems functionalities. In addition, I spent 3 days among the security agents of Brink's at Toulouse-Blagnac Airport. In this section, first I describe the security agents' task by describing their equipment, the prohibited articles, and the protocol to analyze a scan. Next, I highlight their activity issues by showing the different error types, and the dissimulation techniques.

3.2.1 Task description

The airport security agents have many roles depending on their work position. They control people (the passengers, the flight crew, and the airport personnel), and the luggage at security checkpoints. The different work positions are:

- The reception, where the security agents check the documents and put the hand luggage on the treadmill of the X-ray machine,
- viewing and detection of objects and luggage through X-rays,
- pat-down search,
- baggage inspection.

In this study, we focus on the threat detection through X-rays.

3.2.1.1 The Equipment

In most French airport, the security agents two types of fluoroscopic system: the tomograph, and the classic system. The classic fluoroscopic system provides a 2D scan of luggage on the one hand, and on the other, the tomograph can produce transversal scans in addition to the 2D scan. Some scanners are able to detect explosive automatically, they are called EDS (Explosive Detection System). The architecture of the classic system is composed of:

- a mechanical set comprising a frame, a conveyor, and a collimator,
- a physical set comprising an x-ray generator, a detector, and detection cells,
- the central unit,
- and input/output devices (keyboard, screens).

When a luggage get in the frame, the x-rays are sent from the generator to the L-shaped detector through the luggage's components. The x-rays are attenuated depending on the molecular density of the encountered objects, which allows to receive different x-ray intensities on the L-shaped detector. Then, the different received intensities are processed to get the final 2D scan. The final image does not keep the original colors, instead they use 3 colors (orange, green, and blue) to show the density of each encountered object. The orange color is used for low density matters which are mainly organic. In opposition, the blue color is used for high density matters which are mainly inorganic. The green color expresses the superposition of different kinds of materials or average density materials, see Figure 3.1.

FIGURE 3.1: X-ray scan with the 3 standard colors (orange, green, and blue)

3.2.1.2 Prohibited articles

Depending on the context, there are many prohibited articles. The aim is to avoid any kind of weapon to board in the aircraft. According to the regulations, weapons are items intended primarily to kill, injure, immobilize or reduce to impotence. The prohibited articles are:

- Guns, firearms, and any other equipment emitting projectiles,
- Stun devices,
- objects with a sharp tip or a cutting edge,
- working tools,
- blunt instruments,
- explosive or incendiary substances and devices.

In addition to these prohibitions, some articles such as liquid, gels, and aerosols are restricted.

3.2.1.3 Scan analysis

Whatever fluoroscopic system is used, the security agents try to follow an analysis protocol to ensure any prohibited article is detected. This protocol is composed of 4 major steps, which are the global picture analysis, the in-depth analysis, the structure analysis, and the context.

3.2.1.3.1 Global picture analysis

First, the security agent check whether the luggage is well placed. Indeed, the luggage should appear on the screen with 45° of rotation. If it is not the case, a repositioning is required. To avoid this issue, another agent must check the luggage positioning. Next, the agent try to find out whether the luggage is complex or not. A luggage is complex when there is a superposition of usual objects and metallic materials. A complex luggage can hide threat and make them invisible. If possible, the complex luggage is opened and all troublesome objects are removed.

3.2.1.3.2 The in-depth analysis

First, the security agent divides the luggage's component in groups. The aim is to reduce the risk of error. Indeed, smaller prohibited articles may be easily unseen. Then, the dark areas must be lightened up. To do it, the security agent can use functionalities such as high penetration or contour enhancement. If the area is still dark, the luggage must be opened to remove the serious doubts. Next, the agent focus on the problematical shapes. A non-recognizable shape may be caused by a bad positioning. Any unrecognized object must be inspected by the agent. If any prohibited article is detected, the agent follows the procedure related to the threat. In addition, an organic matter superposed to an electronic device is considered as a sufficient reason to open the luggage. Liquids, gels, and aerosols should be less than 100ml otherwise they will be removed from the luggage. These liquids should be placed into a plastic bag.

3.2.1.3.3 The structure analysis

During this step, the security agent compare each element of the luggage which must be symmetrical because of its industrial conception such us suitcase's wheels. If the symmetry is not respected, it is might be a clue to the deliberate luggage modification for dissimulating prohibited objects.

3.2.1.3.4 The context

Depending on the context some articles are prohibited or not. The agent must be aware of his work position. On an explosive detection system, the threats are already highlighted. The agent's job is to verify whether the threats are real or not. In our study, we focus on a conventional system to provide new interaction techniques for analyzing 3D luggage scans.

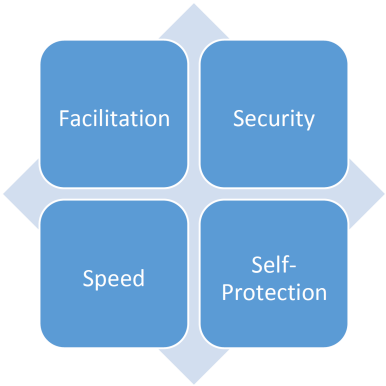


FIGURE 3.2: The operational constraints of the security agents.

3.2.2 The activity issues

There is many level of security in luggage inspection depending on the airport (between 1 and 6). Humans and machines work together to detect any potential threat inside the luggage. At the lowest levels of security, the security agents are in contact with the passengers, and therefore face 4 main operational constraints for safety and economical reasons: Facilitation, security, speed, and self-protection.

First of all, The facilitation constraint implies that the security agent has to ensure a good user experience to the customers (the passengers). Second, they must prevent any potential threat to pass their inspection. In addition, they have to protect themselves from different kind of threats (ill-intentioned persons, parcel bomb). And finally, economical reasons oblige them to be fast enough in order to ensure a good flow. Therefore, the security agent face a trade-off between these four constraints, see Figure 3.2 .

3.2.2.1 Error Types

The security agents can usually do 3 types of error during the baggage inspection process.

3.2.2.1.1 Perception errors

The security agent does not detect the right area to analyze. He is not focus on the troublesome area, see Figure 3.3.

3.2.2.1.2 Interpretation errors

The security agent detects the right troublesome area but does not judge it as a potential threat. In other words, he is focused on the right area but cannot see the menace.

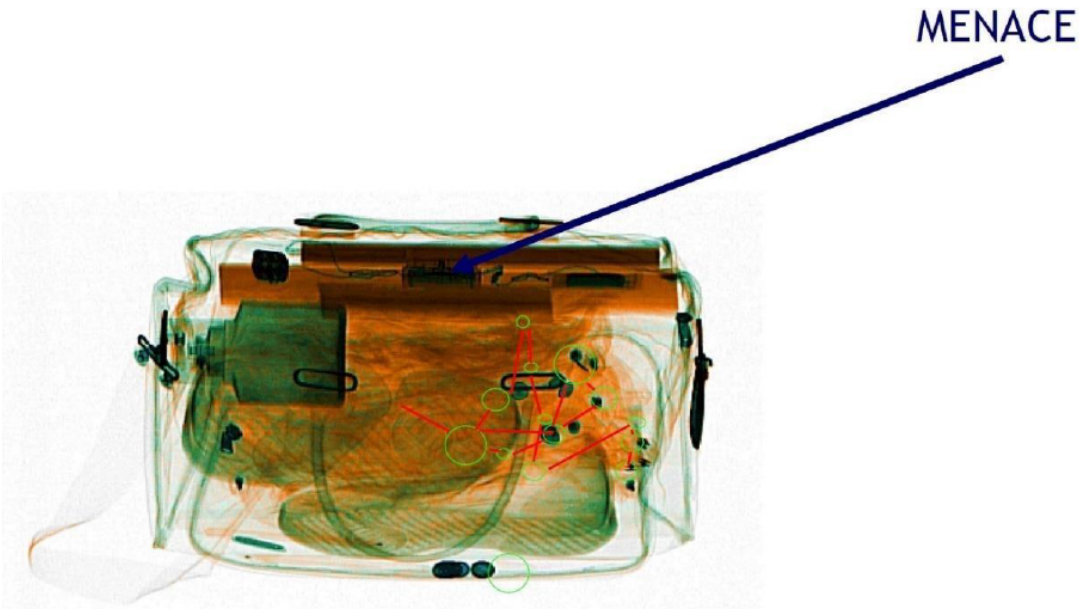


FIGURE 3.3: Perception error. The security agent do not notice the menace at the top of the image (the initiating system of an explosive)

The agents used to combine what they find out with their knowledge on real world objects stored in their memories. They have their own model of the situation, see Figure 3.4.

3.2.2.1.3 Decision making errors

In this case, the security agent detects the right troublesome area with a good interpretation, but make a bad decision. It might be related to the context or the procedure knowledge. In addition to these errors related to the agent, four hiding techniques can be used to bypass the security mechanisms. Those techniques are superposition, positioning, dissociation, and bait.

3.2.2.2 Dissimulation techniques

Since the resulting X-ray scanned image only contains densities, it cannot display the material original colors. The standard color visual mapping uses 3 different colors (orange, green, and blue) to display the data density. Orange color corresponds to low density (mainly organic items). In opposition, blue color is used for high density items (e.g. metal). In the case of X-ray systems, green color corresponds to the superposition of different kinds of materials or average density materials (Figure 3.1).

The displayed 2D scanned image can suffer from four issues.

Superposition: A threat (e.g. prohibited object like knife, cutter...) may be sheltered behind dense materials. Sometimes, it is possible to see through these blind

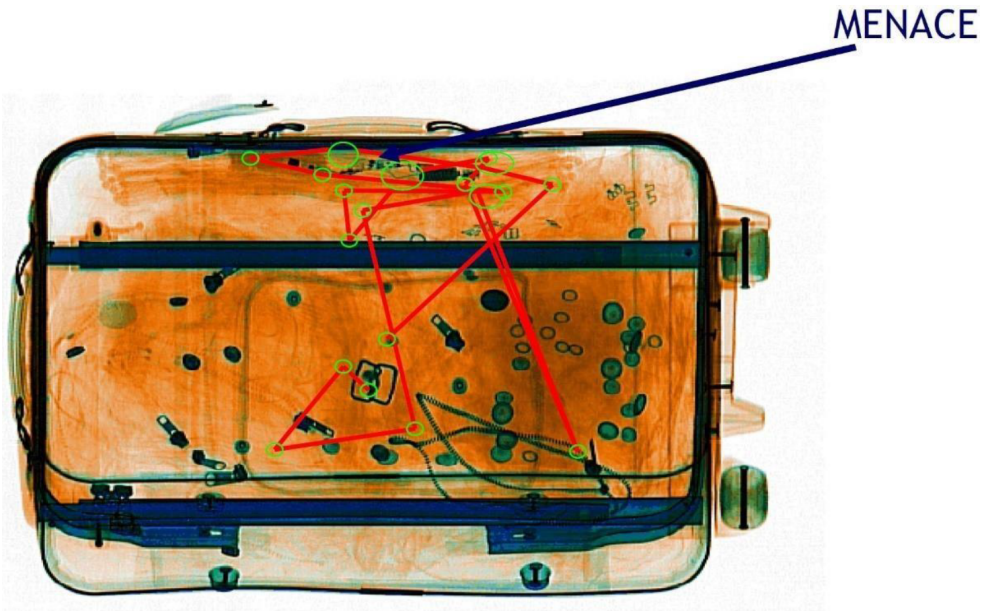


FIGURE 3.4: Interpretation errors. The security agent notice the menace at the top of the image do not interpret it as a threat (the initiating system of an explosive)

shield using some functionalities such as high penetration (enhanced X-ray power) or image processing (contrast improvement), see Figure 3.5.

Location: Depending on its location inside the luggage, a threat can be difficult to detect. Objects located in the corners, in the edges or inside the luggage's frame are very difficult to identify, see Figure 3.6.

Dissociation: Another way to dissimulate a threat is to separate and to spread parts of it in the luggage (weapon or explosive are composed of many separated items like the trigger, the cannon...). This dissociation can be combined with other dissimulation techniques, see Figure 3.7.

Lure: An ill-intentioned individual may use a lure to hide the real threat. For instance, a minor threat like a small scissors may be clearly visible and catch security agent's attention while a more important threat remains hidden, see Figure 3.8.

3.2.3 Requirements

3D baggage scan exploration are one potential solution of such limitations, but to the best of our knowledge no previous existing system investigated this activity domain with interactive volumetric exploration tools. Even if extensive works have been done in medical 3D scan exploration and manipulation Preim and Botha, 2013, there is a great opportunity to adapt and develop new interaction and data manipulation techniques to support 3D baggage exploration.

We performed on site observations with contextual inquiry (one day observation

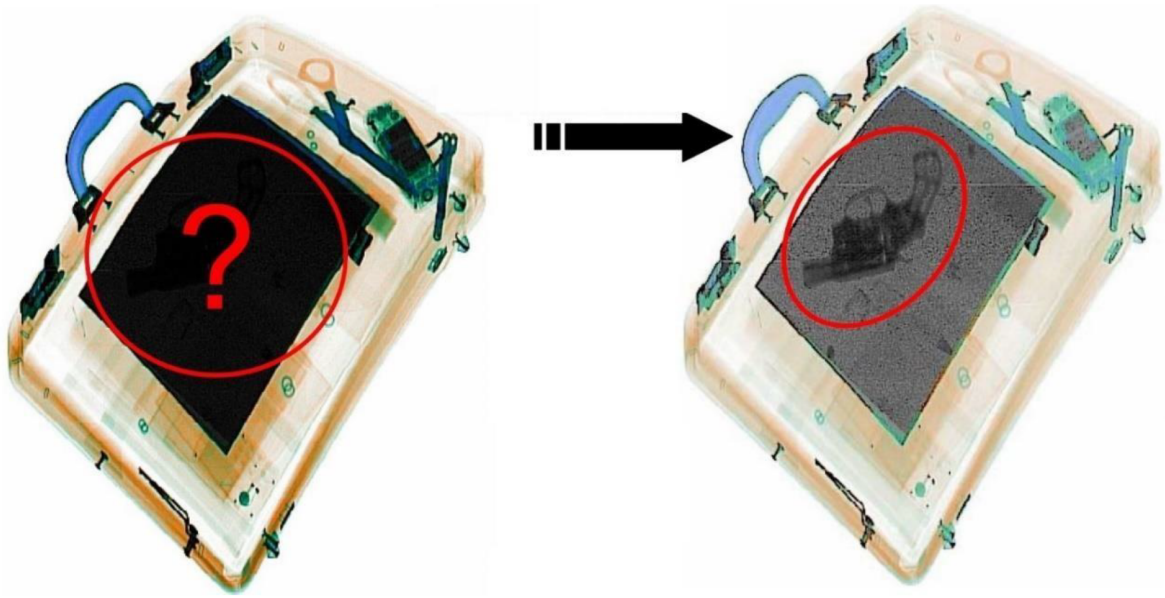


FIGURE 3.5: Dissimulation by superposition

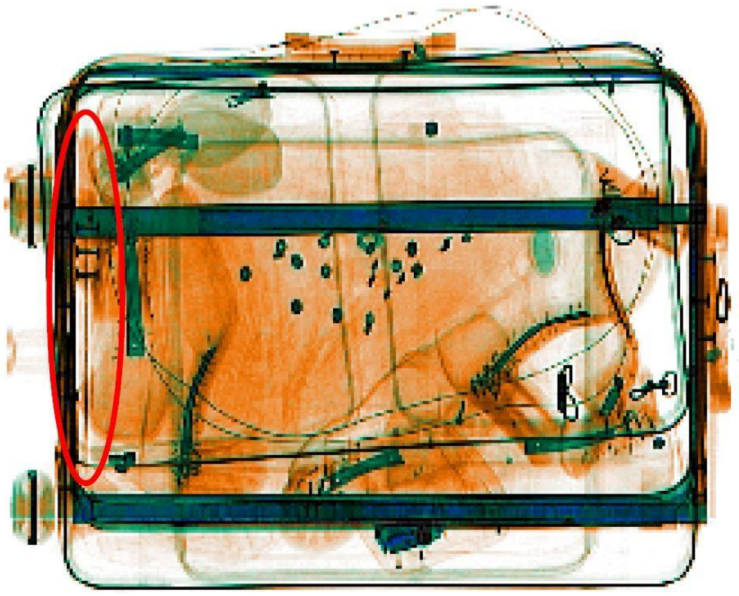


FIGURE 3.6: Dissimulation by the location

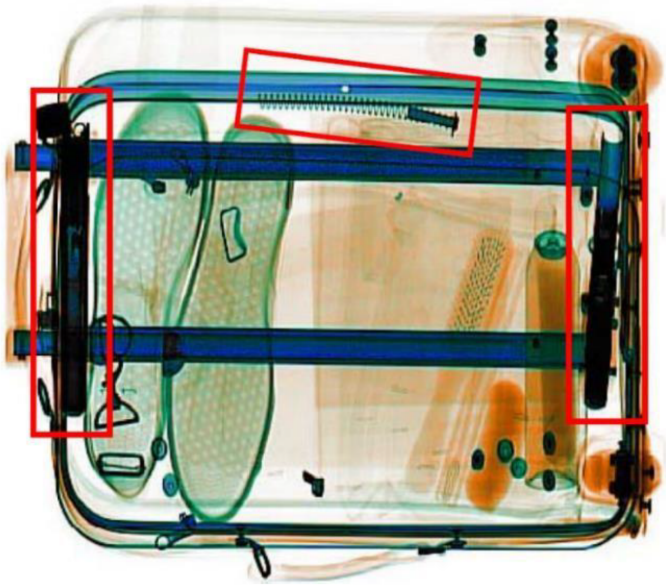


FIGURE 3.7: Dissimulation by dissociation

FIGURE 3.8: Dissimulation using a bait

in one of the major French airport). We also conducted one brainstorming with four security practitioners from which we defined relevant use cases. Thanks to these analysis, we extracted a set of top level needs and requirements :

- VIS: users need to visualize the content of the luggage.

- EXP: users have to explore the baggage with interactive navigation system.

- OCL: the system must provide tools to address occlusion issues; the superposition of items inside the luggage hinders their visualization.

- INT: User need to explore luggage with simple interactions. User knowledge is too limited regarding volumetric data processing to understand the involved techniques and their parameters.

3.3 Interactive exploration of 3D scanned baggage

3.3.1 Top Level Structure

Our system is composed of one main view (Volume Visualization) and 5 sub views to control and customize it, see Figure 3.9. Our interactive system does not provide menu and every feature is directly accessible from this main view (Figure 3.9). The Volume Visualization can contain up to two views to ease interactions with the 3D scan. These views display the baggage and one can navigate through it (zoom, pan, rotation), manipulate its content (brushing, selection, deletion). These two views can be linked or disconnected in order to inspect selected objects with or without their context. A smaller view called Overview shows the location of the investigated area. The overview is one-eighth the size of the main window and is located at its bottom right.

Temporal Instances window, located in the top right of the interface, shows the current and the previous settings of the Volume Visualization. The current setting is modified when the user changes the transfer function or manipulates an object. The Snapshot shows saved instances with their setting (pan, zoom, rotation, transfer function, selection, brushing). The Toolbox contains every interactive tools to explore the baggage (brushing, selection, eraser, density picker, navigation tools). Transfer Function presets contains six predefined settings ordered by their filtering power. Low level shows every density, high level only show high density values.

This global layout can be customized thanks to View Options. One can display one or two views of the Volume Visualization, the Temporal Instances, the Snapshot and the Overview.

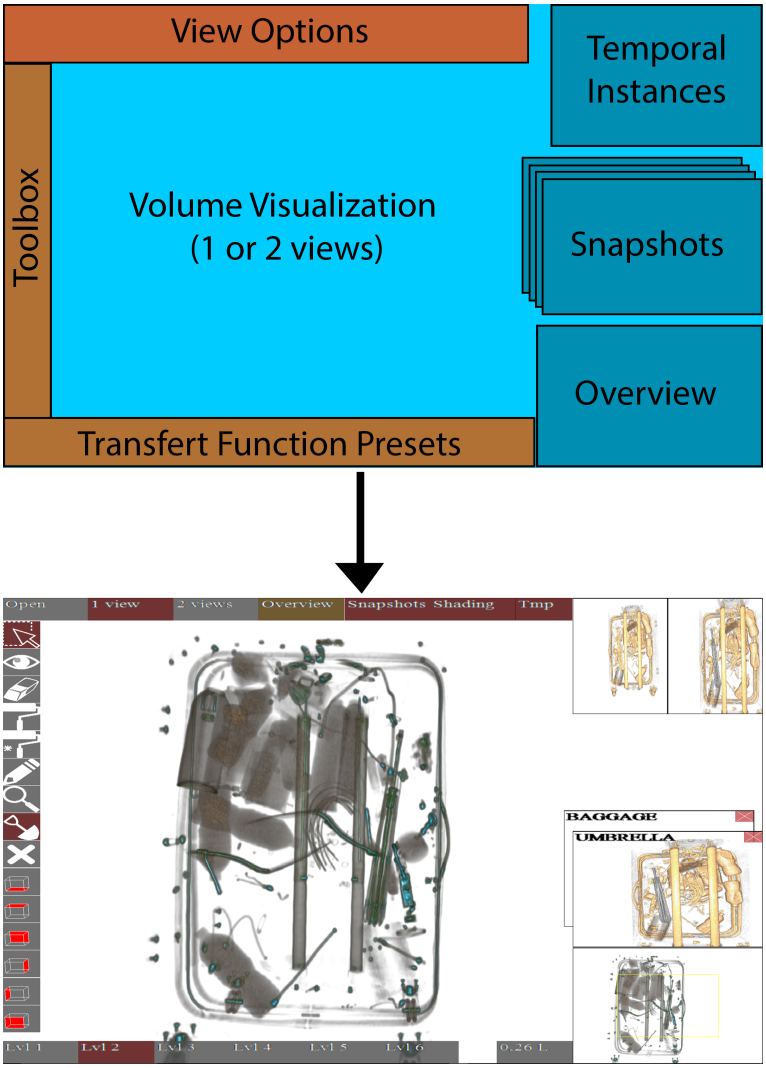


FIGURE 3.9: Top level layout and screenshot of our Graphical User Interface. All the features are available through this GUI

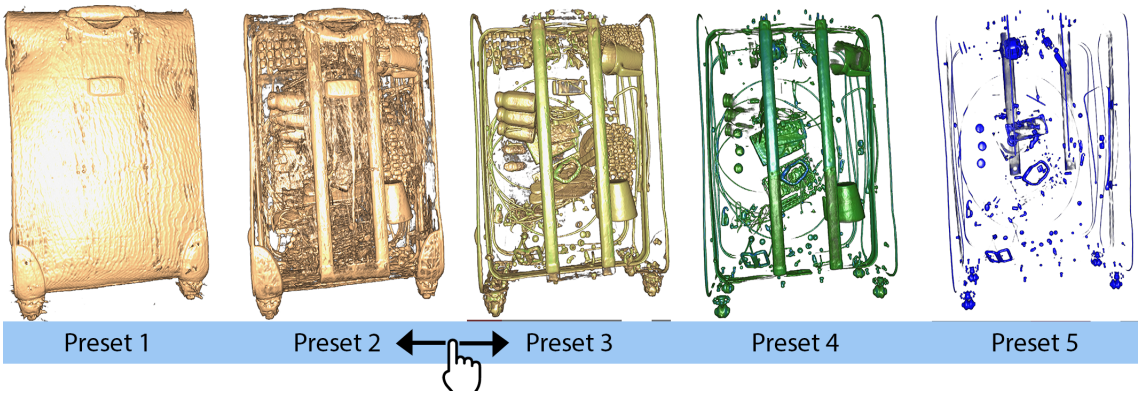


FIGURE 3.10: Transfer function presets and their continuous interaction. The high density materials are revealed by dragging from left to right on the presets

3.3.2 Interaction techniques

We used a multi touchscreen Wacom 24HD equipped with a stylus. Most of the developed interactions can be performed with every available modality: mouse, hand or pen.

3.3.2.1 Transfer function edition

Datasets such as two dimensional raster images or three dimensional voxel based representations are often processed for representation using a transfer function (TF) defined by a curve.

Since airport security agents have a reduced time frame and limited knowledge of technical constraints, we defined six TF presets (Figure 3.10). These presets only modify the TF transparency curve while keeping the same color mapping. These presets are ordered by their density filtering power: the first preset displays every density and the last one only highest density of the volume (i.e. metal).

When clicking on a preset, the TF changes with a short transition (1 second) toward a new one. Finally, the user can select a transfer function located between two presets. To do so, the user drags from any location within the TF presets area until the volume visualization shows interesting features

3.3.2.2 Objects selection and investigation

In order to investigate in detail a specific object, one can isolate it, remove surrounding items to address occlusion issues, or find a suitable point of view (Figure 3.18). This section details these interaction techniques which encompass the selection of many objects.

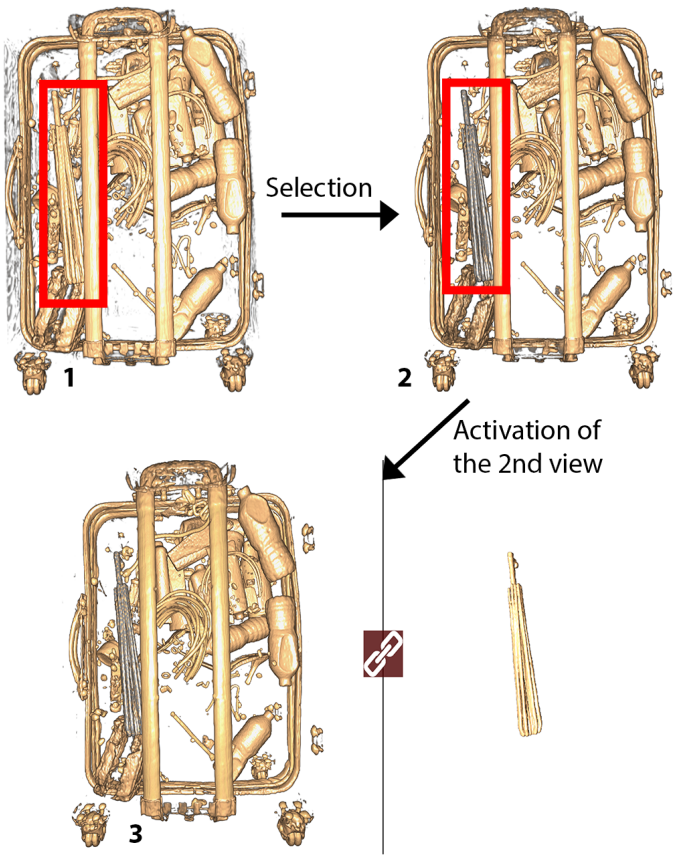


FIGURE 3.11: Selection of an umbrella for further inspection. 1- The user wants to check an object looking like an umbrella. 2- After a double click, the object is selected and becomes gray as a feedback. 3- After the activation of the 2nd view, the user can manipulate the umbrella with or without the whole baggage. All the Selected items are also available on this second view.

3.3.2.2.1 Selection of an object

We developed three different ways to select objects with the three different modalities: hand, mouse or pen. When using his or her hand, the user has to double tap the desired object with his or her finger. When using the pen, the user must press the biggest button on the stylus while pointing at the target. Otherwise, the user has to double click on the target with the mouse pointer. After the target is selected, our system tries to find a better point of view to display the selected object with the minimum of occluded parts. The details of this algorithm will be explained in the technical part of this study. We use a smooth transition to rotate and zoom the baggage.

If the user has activated the 2 views Volume Visualization (Button 2 views in the View Option panel), the selected object is isolated in the second view (on the right side of the main view Figure 3.11). The selection is incremental and many object can be selected one after another. To undo the selection, the user has to select the desired object in the second view.

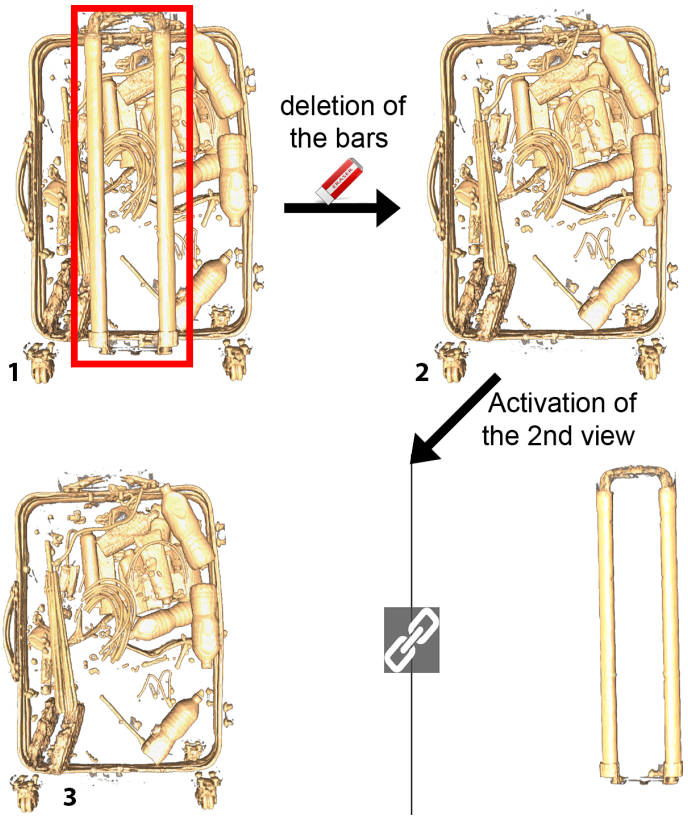


FIGURE 3.12: Addressing occlusion issue by removing objects from one view to another one. 1- The user wants to remove the bars of the baggage. 2- After a double click with the deletion tool, the object is removed from the main view. 3- After the activation of the 2nd view, the removed objects are visible outside the baggage.

3.3.2.2.2 Occlusion management

In order to address occlusion issues, the eraser tool can interactively remove selected objects from one view to the other one. This interaction is similar to the selection interaction with all the modalities (mouse, hand, and pen). When using his or her hand, the user has to double tap the desired object with his or her finger. When using the pen, the user must press the biggest button on the stylus while pointing at the target. The user can also double click on the target with the mouse pointer. To restore a deleted object from one view, the user has to erase it from the other view. This simple principle guarantees that every item of the baggage is always visible while addressing occlusion issues (Figure 3.12).

3.3.2.3 Extended Brushing techniques

Baggage is composed of low density items (e.g. fabrics, clothes, papers, organic objects...) and high density ones (e.g. metal, electronic components...). Low density items are more numerous and surround high density ones. In order to perceive a metallic object, one needs to remove or hide its surrounding low density items thanks to an adequate TF. Since TF is applicable to the whole volume visualization,

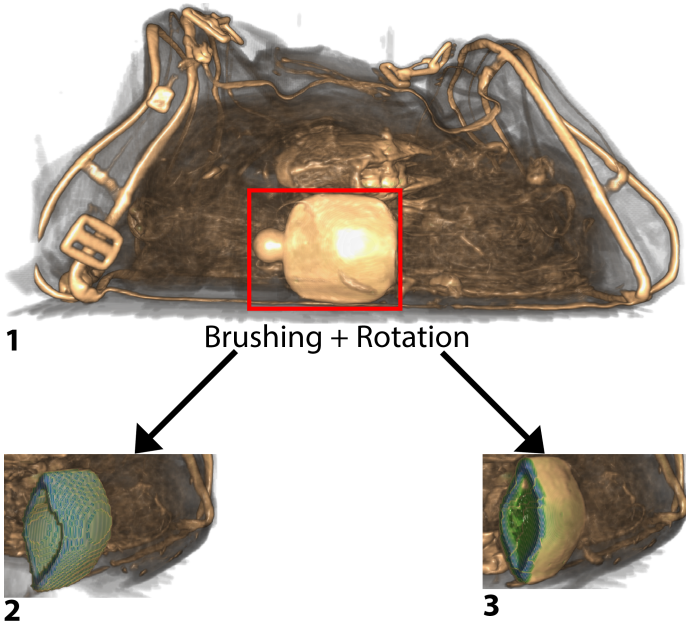


FIGURE 3.13: Brushing a bottle to see its content. 1- The initial baggage before brushing. 2- The result after using the old brushing technique: the content of the bottle has been removed too. 3- The result after using the new brushing technique: the content of the bottle is still visible. This brushing technique removes until it encounters an object out of the selected density range.

the visualization of a metallic bottle will prevent to visualize its content (i.e. organic density). One cannot find out whether the metallic bottle is empty or not (Figure 3.13). To solve this problem, we developed two brushing techniques to explore the baggage by the mean of local removal of specific densities without modifying the TF.

3.3.2.3.1 Range Based brushing technique

This interaction removes all voxels which are bounded in a user defined range. A range-slider is available on top of the transfer function (Figure 3.14). The user enables the brush tool in the toolbox by clicking on it and then the right button of the mouse removes the voxels beneath the mouse pointer. Original brushing technique removes every voxel which is bounded in a density range and which is beneath the mouse pointer. In case of the exploration of a metallic bottle full of liquid, original brushing technique will remove the surrounding of the bottle as well as its content. To address this issue we developed a new brushing technique as if one digs into a baggage. This digging process will stop when a dense layer is encountered. This dense layer (e.g. the outer layer of a metallic bottle) will act as a shield to protect voxels located behind it (Figure 3.13).

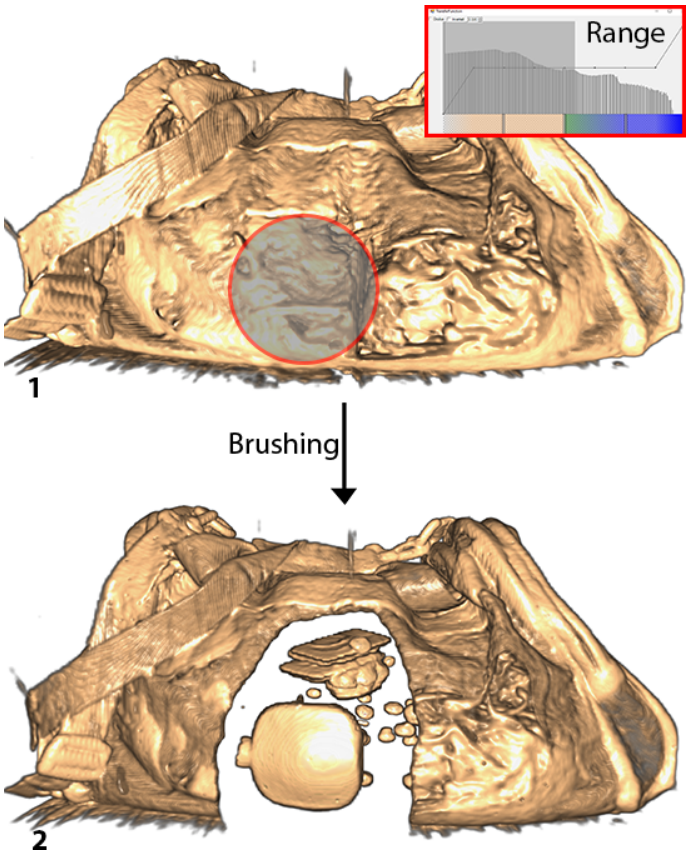


FIGURE 3.14: Brushing low density materials to see a metallic object hidden inside the baggage. 1- The baggage before brushing. 2- After a density range is defined, the users brushes a part of the baggage. This interaction reveals a metallic bottle.

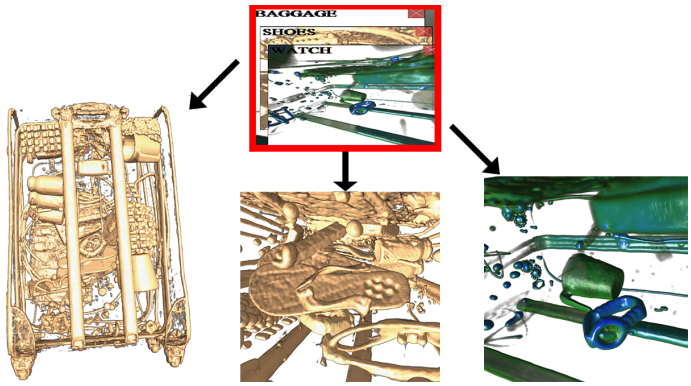


FIGURE 3.15: The labelled snapshots taken by the user representing objects of interest

3.3.2.3.2 Magic brushing technique

This brushing technique has almost the same behavior as the range based brushing one. The main difference consist in the automatic range definition. This magic brushing removes all voxels with a lower density than the first one encountered at the beginning of the brushing process. This technique helps the user to directly define the densities he or she wants to brush. This technique avoid multiple interactions with the histogram and its range slider to define the range of brushable densities.

3.3.2.3.3 The cancellation of the brushing

Our system offers the possibility to restore previously brushed areas. To do so, the user must select the brushing tool in the toolbox and hold the shift key while brushing over a given area. The restored densities are those defined by the histogram range slider. This restoring processing does not act as the range brushing technique and the sheltering effect (dense layer provides to brush behind it). Every voxels within the range are restored in order not to confuse the user.

3.3.2.4 Snapshots

Our system can store visual configurations thanks to snapshots. These snapshots are displayed on the right of the main view and record the current rotation, the zoom, the pan and the current transfer function. To take a snapshot, the user must press the space bar. A name can be given to any snapshot using the keyboard. This name is displayed on the top left of its thumbnail. The user can restore a saved object by clicking on its snapshot. When a snapshot is selected, the system animates the view from its current state to the selected snapshot (pan, zoom and transfer function linear interpolation). A snapshot can be deleted by clicking on the close icon located on its top right (Figure 3.15).

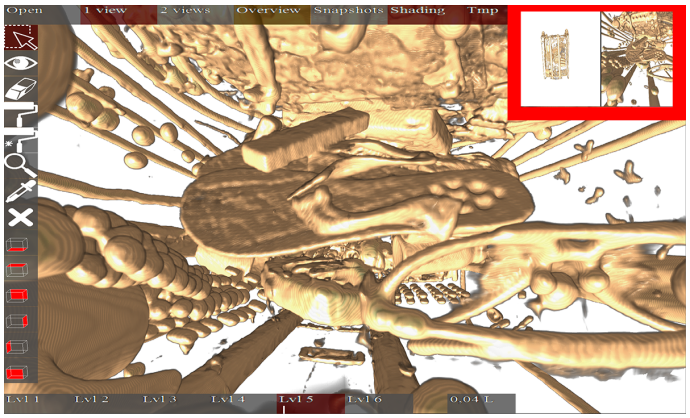


FIGURE 3.16: The labelled snapshots taken by the user representing objects of interest

3.3.2.5 Dual temporal instance navigation

Our system provides many animated transitions when the user explores the baggage (change of point of view, transfer function modification, item selection...) like Tversky, Morrison, and Betrancourt, 2002. To ease user's navigation, we added two temporal instances representing the states before and after an automatic modification of the visual configuration. This interaction is based on the undo-redo paradigm. On the top right of the main window, we added two views representing the previous state before the transition and the final one after the transition. One can navigate through this transition by dragging from one state toward the other one. The more the cursor gets close to the center of a state, the more the current view gets close to its configuration. The user can also click on the desired state to directly assign its visual configuration (Figure 3.16).

3.3.3 Possible Use Cases

This section presents two scenarios that underline our system assets. The first scenario illustrates how users can explore a suspicious baggage and address occlusion to inspect items inside. The second scenario shows how to visualize the content of a closed metallic container. Currently, most airport don't use 3d systems, so we propose possible scenarios to highlight how the usefulness of our interaction techniques.

3.3.3.1 Inspection of a suspicious baggage

When a baggage arrives, the user sees it in its initial appearance which mainly shows the global structure . In order to better perceive its content, the user explores the different transfer function presets. He or She clicks on the first level, then drags the cursors toward the others. This manipulation aims to reveal high density items hidden by those with low density

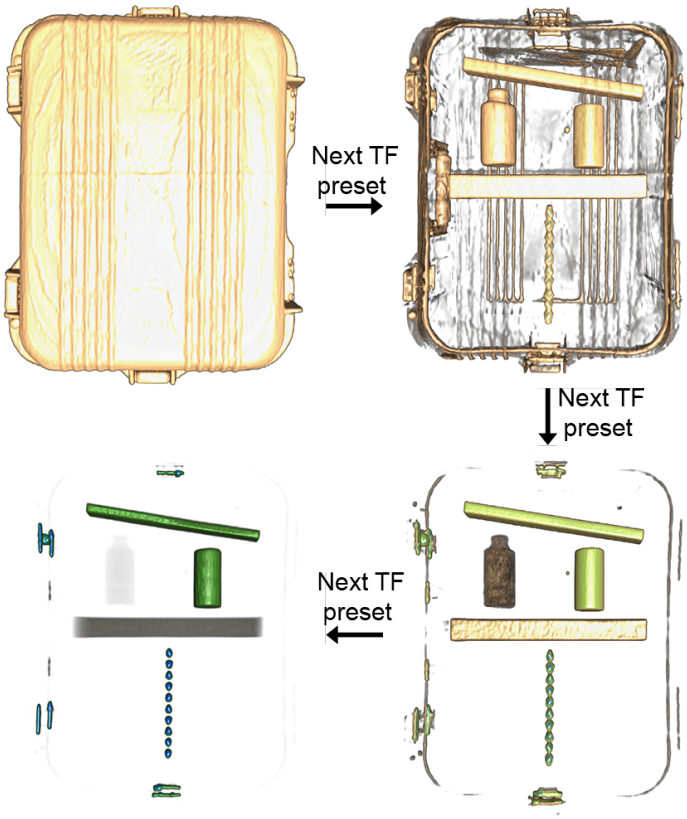


FIGURE 3.17: The transfer function presets help the user perceive the different types of material inside the baggage

(Figure 3.17). Once the user has a global understanding of the baggage, it seems to be a parcel bomb. So he or she decides to further inspect some specific areas. For instance, the user focuses an object which looks like an explosive material. To estimate the potential damages that could be caused by this potential threat, the security agent lock the camera on it. This object is now the main object of interest, and all manipulations (e.g. translation, rotation) are now done around this object. This object of interest stays visible even if other items are in front of it (Figure 3.18).

After the user finished this contextual inspection, he or she decides to inspect this object alone. To do so, the user selects it, and enables the second view. More information such as density and volume are displayed (Figure 3.19).

3.3.3.2 Inspection of a big metallic object

In this scenario, the user inspects a baggage featuring many visual artifacts which could be considered as noise (Figure 3.20). These spikes are created by the reflection of the x-rays on the heavy materials. Therefore, these visual artefacts are a hint indicating the presence of large metallic object. The user navigates through the different transfer function presets (Figure 3.21) .

After the selection of the adequate transfer function setting, he or she decides to farther inspect this metallic object. To do so, the user selects it with a double clicking

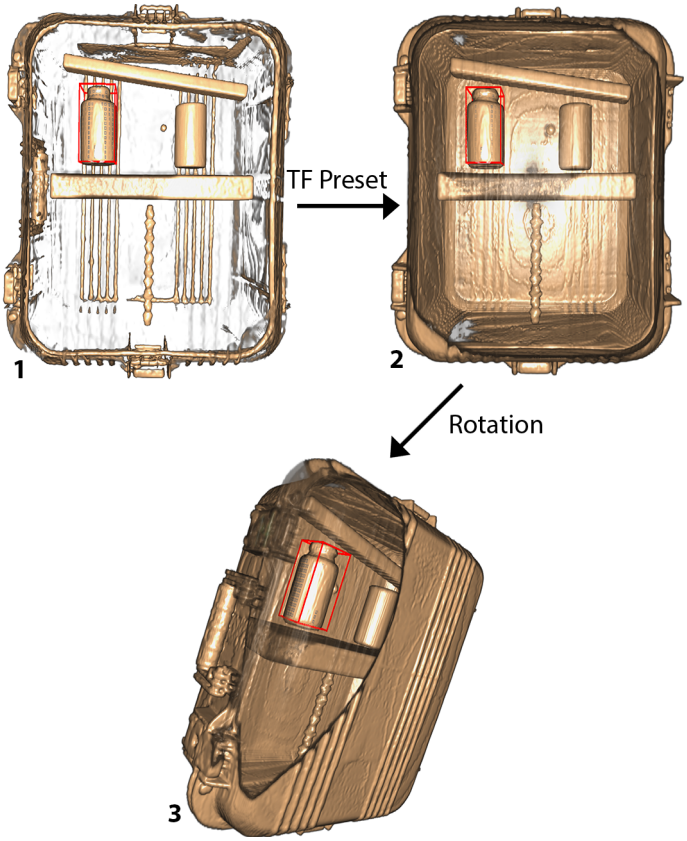


FIGURE 3.18: Inspection of an object from different perspectives. 1- The user defines an object of interest for further inspection. This object is then inside a box with red borders. 2- The user modifies the transfer function to see the target in another context. 3- The user rotates the baggage to look at the menace from a different perspective. It stays visible whatever the manipulation carried out.

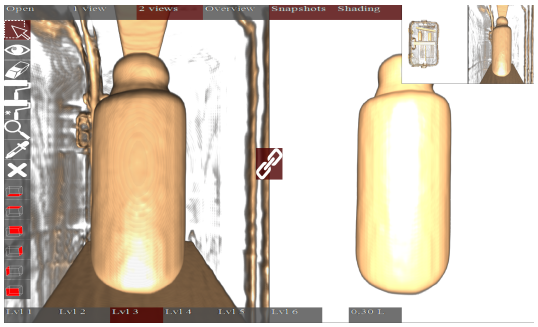


FIGURE 3.19: A suspicious object is inspected apart of the rest of the baggage

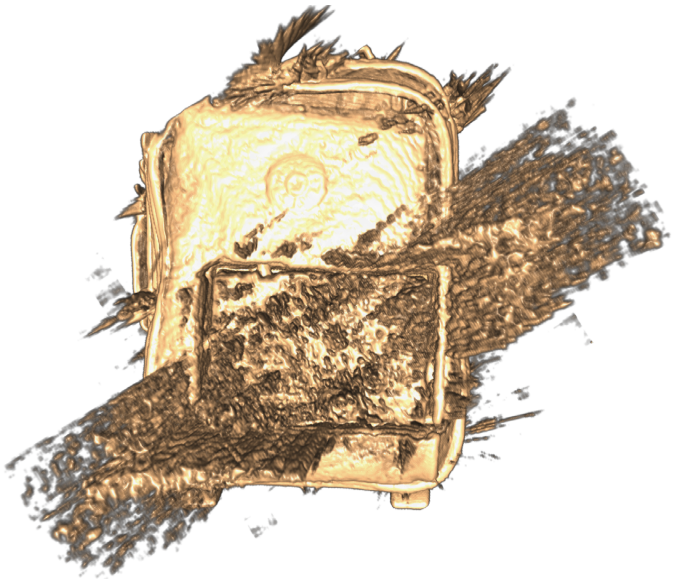


FIGURE 3.20: Noise due to the presence of a big metallic object

on it. This object seems to be an engine.

In order to find out whether this engine is empty or not, the user decides to brush its half and to see its content. The user activates the brush tool, and while holding down the right button of the mouse, he or she brushes to remove the undesired section. After the brushing is carried out, the user modify the transfer function to show the lowest densities(Figure 3.22).

3.3.4 Technical Constraints and Implementation

Tomographs generate files which contain all the densities encountered by the x-rays through the baggage. These files usually contains around 30 million values. To load and to display such dataset in an interactive system, we used the parallel computation power of the graphic card. We used a program written in C# combined with the CUDA parallel computing platform and a set of GPGPU techniques. To display the dataset, we used a standard ray casting algorithm. The contributions of you work rely on new interactions and their associated algorithms. This section will detail their implementations.

3.3.4.1 Object selection

Our software features interactive selection and voxel manipulation functionalities. To implement these interactions, we had to find a way to extract objects from the volume data. To do so, we explored many object detection algorithms such as contour trees by Carr, Snoeyink, and Axen, 2000 and branch trees by Pascucci, Cole-McLaughlin, and Scorzelli, 2004 algorithms. But, according to the literature and our

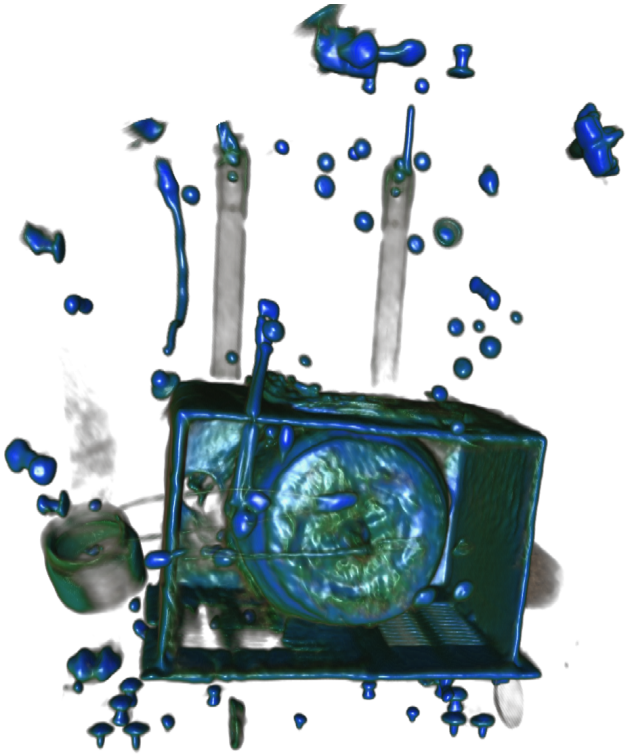


FIGURE 3.21: The baggage actually contains an engine

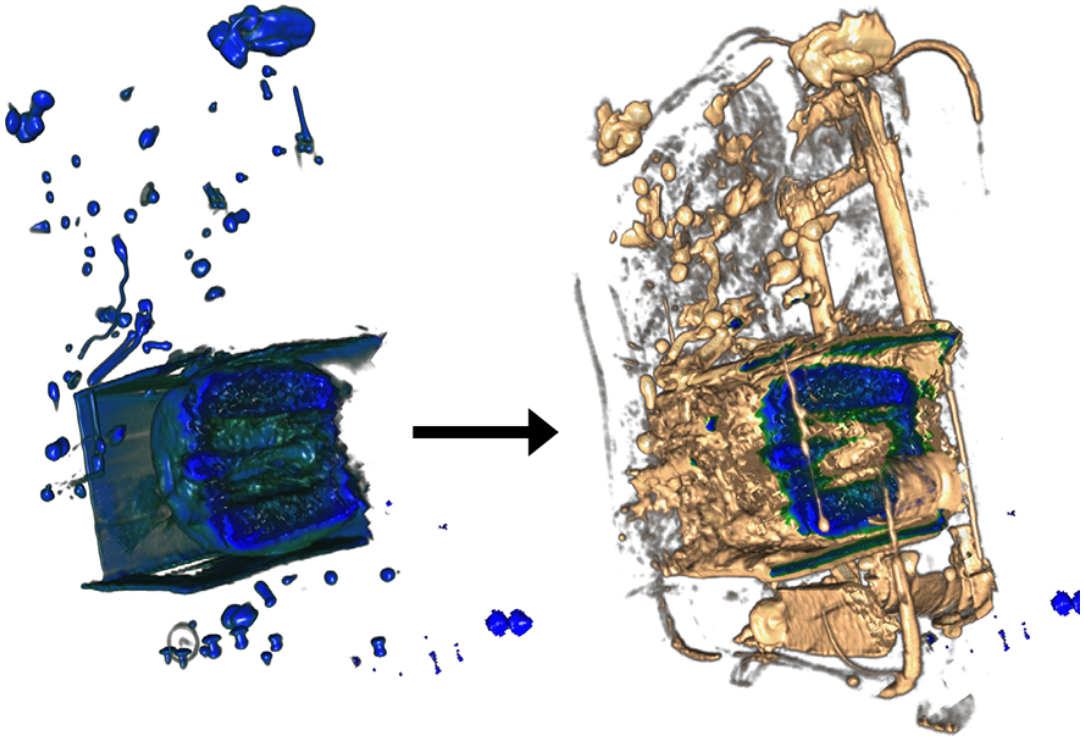


FIGURE 3.22: the user reveals some low density materials inside the engine

experience, these algorithms are computationally expensive and time-consuming especially during the preprocessing steps. In addition, this algorithm needs many settings and modifications to process the variety of available possible items which compose a baggage. For these reasons, we developed our own selection algorithm. It is based on a simple brute force multi-threads propagation algorithm. The first step consists in casting a single ray toward the volume data where the selection need to start (we stop its progression as soon as it hits a visible voxel). Second, we propagate the selection in every directions. For each encountered voxel, we check whether it is visible according to the current transfer function . If the voxel is visible we go on with the selection propagation. We spread within the baggage by launching many threads starting from the location where the first visible voxel has been hit. The number of thread depends on the number of logical processors inside the current hardware. Each thread checks whether its neighbors are visible, and keep spreading towards the visible voxels. This propagation algorithm stops whenever there are not any more visible voxels to spread into.

3.3.4.2 Occlusion minimization

Once the item of interest is selected, our system animates the visual configuration to display a new point of view of the selection which minimizes object occlusions. To do so, the system first computes the selection bounding box composed of six faces. Then, the system computes the six possible point of views. For each of them, the system counts the number of visible pixels which display the selected object. This is done thanks to the ray casting algorithm which propagates though the data cube and thus can test if the ray hits the selected items without any occlusion. Finally, the system defines the appropriate point of view as the bounding box face with the highest number of visible pixels of the selected item. The system can then animate the volume visualization toward this new visual configuration and update the two temporal instances (before and after the pan, zoom and rotation modifications).

3.3.4.3 Extended brushing technique

The extended brushing techniques help the user explore the volume without modifying the transfer function. This interaction is also based on the ray casting algorithm. We only cast the rays located inside the radius of the brush in a parallel way. When a ray hits a visible voxel whose density is within the brushing range, this voxel is then removed. If the voxel density is not in the brushing range, the ray casting algorithm stops. The cancellation of the brushing is quite similar to the previous algorithm. The rays start from the back of the volume toward the front instead of front-to-back. The second difference consists in restoring the voxels whose density is inside the range.

3.3.5 Discussion and conclusion

Since we developed this system with four baggage security practitioners, we had the opportunity to validate the usefulness of each developed technique. Through an interactive process, the users gave their feedbacks all along the development process which helped to assess and to guide the proposed interaction techniques. Feedback was mainly positive and the user did not face any specific difficulty to use our system. Users mainly appreciated the simple interface with few widgets and the reduced set of interactions. Surprisingly, users were very interested to use the transfer function. The histogram and its transfer function were also appreciated even if the corresponding technique is not simple to understand. We suppose that our interface motivates users to learn more regarding the technique behind it. Users also mentioned the need to display the actual density (the numerical value) of selected object. They also ask many times if the displayed color corresponds to the one currently in use in operational settings. This confirms the fact that users are willing to keep some existing features and prefer to use a system they are already familiar with. The users also appreciated our design requirements with smooth transitions and incremental investigations of baggage.

All of these observations and qualitative evaluations deserve to be validated through proper evaluations which are out of the scope of this user study paper. Our system is fully functional and interactive enough to perform baggage explorations. Many improvements can still be done in order to improve its usages in terms of new interactions and exploration time reductions. Technically speaking, the selection of the adequate point of view can be improved with more than 6 investigated faces. But in practice, this simple paradigm remains suitable. If the computed point of view is not fully satisfactory, one can manual rotate the baggage. Nevertheless, the developed algorithm will not necessary provide the best solution (the lowest occlusion) but a satisfactory one.

Our goal was to develop innovative interactions to support baggage exploration but we did not really took into account the optimization of the exploration duration time. Manipulating the 3D volume may take time as well as our new interaction techniques. We think that these techniques have a great potential but they are suitable to explore in more details a suspicious baggage. Existing investigation techniques (with the 2D flattened image) are suitable to quickly and efficiently detect ”clean” baggage. Then our tools can be a good solution to further investigate a potential threat with more available time.

Chapter 4

Interactive obstruction-free lensing for volumetric data visualization

Occlusion is an issue in volumetric visualization as it prevents direct visualization of the region of interest. While many techniques such as transfer functions, volume segmentation or view distortion have been developed to address this, there is still room for improvement to better support the understanding of objects' vicinity. However, most existing Focus+Context fail to solve partial occlusion in datasets where the target and the occluder are very similar density-wise. For these reasons, we investigate a new technique which maintains the general structure of the investigated volumetric dataset while addressing occlusion issues. With our technique, the user interactively defines an area of interest where an occluded region or object is partially visible. Then our lens starts pushing at its border occluding objects, thus revealing hidden volumetric data. Next, the lens is modified with an extended field of view (fish-eye deformation) to better see the vicinity of the selected region. Finally, the user can freely explore the surroundings of the area under investigation within the lens. To provide real-time exploration, we implemented our lens using a GPU accelerated ray-casting framework to handle ray deformations, local lighting, and local viewpoint manipulation. We illustrate our technique with five application scenarios in baggage inspection, 3D fluid flow visualization, chest radiology, air traffic planning, and DTI fiber exploration.

4.1 Introduction

Direct volume rendering (DVR) is a pervasive visualization technique for displaying 3D scalar fields with applications in engineering, material sciences, and medical imaging sciences. However widely adopted, and able to handle large datasets at interactive rates, DVR inherently suffers from the problem of *occlusion*: Structures of interest located deep in the volume, called next *targets*, can be hard to spot and/or explore.

To aid with this, various techniques have been designed including transfer functions, segmentation, selection, and clipping. Yet, all such techniques have limitations. *Global* mechanisms, like transfer function editing, can remove both occluders

and targets if these have similar densities. In certain applications, carefully designed transfer functions exist and should be used without (significant) modifications to facilitate understanding and user training Wu and Qu, 2007. *Local* mechanisms like segmentation, selection, or clipping are more effective in manipulating data confined to a given spatial region. Yet, many such mechanisms assume that one can easily and accurately select targets to remove them (occluders) or keep them (occluded). This is hard to do when *e.g.* one does not have direct access to the targets, or when significant 3D interaction is required to select occluder(s).

A different way to handle occlusion is to use *lenses*. These are flexible lightweight tools which enable local and temporary modifications of the DVR to reveal targets while keeping the global visualization context, Carpendale, Cowperthwaite, and Fracchia, 1997; Tominski et al., 2016; Tominski et al., 2012. However, efficiently selecting the target and removing all in-between occluders is still challenging. More specifically, most existing occlusion management techniques do not simultaneously meet all following requirements: Rapidly create an unobstructed view of the target (R1), allow a flexible local exploration of the target zone (R2), keep the context in which the target is visually embedded (R3), and handle datasets where the target and occluders cannot be separated by transfer function manipulations (R4).

In this chapter, we increase the flexibility of lenses for DVR exploration to jointly cover all above requirements. We propose a focus-and-context (F+C) lens that combines a distortion technique, which pushes aside the occluding objects, with a fish-eye field of view, to provide a better perspective on targets. We specifically target the use-case of *partially occluded* objects, where the user has a glimpse of an interesting structure, buried deep in the data, but only slightly visible from a given viewpoint and transfer-function setting. We allow the user to "open up" the volume without changing these settings, and reveal the target, by simple point, click, and scroll operations. Next, we provide several F+C modifications of the lighting parameters, transfer function, and geometry in the focus area to better understand the target. Our technique, implemented using a CUDA-based approach, can be easily incorporated in any generic DVR system.

The chapter is structured as follows. section 4.2 presents four requirements for occlusion management in DVR visualization. section 4.3 introduces the principle of our lens. section 4.4 introduces implementation details.section 4.5 presents five application scenarios for our lens in baggage inspection, 3D fluid flow visualization, chest radiology, air traffic planning, and DTI fiber exploration. section 4.6 discusses our proposal. Finally, Section section 4.7 concludes the chapter.

4.2 Requirements

To start with, let us refine requirements R1, . . . ,R4.

- **R1:** The technique should rapidly create an unobstructed view of the target, *i.e.*, such a view should be created at interactive frame rates (10 fps or more) with no special pre-processing of the input volume (*e.g.*, segmentation), and

with minimal user input (*e.g.*, using simple mouse and/or keyboard-modifier events). All above are needed to ensure that one can freely *explore* the volume by activating the lens anywhere with minimal effort and seeing its effect in real-time.

- **R2:** Allowing a flexible exploration of the target zone means that one can manipulate the zone in the lens in various ways to see how the target is actually embedded in its surrounding context.
- **R3:** Keeping the context means that the visualization around the lens does not change significantly from what would be shown there if the lens were not activated. This is needed to maintain the user's mental map before *vs* after activating the lens.
- **R4:** The lens should enable the exploration of datasets where targets cannot be easily separated (isolated) from their surrounding context simply by manipulating parameters of the transfer functions (TF). One such issue is when targets and surrounding zones have similar densities; in this case, using a single global opacity TF could either render everything opaque (thus, it will be hard to visually isolate the target) or highly transparent (thus, the zone around the target will be transparent but so will be the target too).

Previous work on handling occlusion when exploring volume data can be divided into occlusion management techniques and lenses-and-deformation techniques. We next discuss these and also point out limitations from the perspective of R1, . . . ,R4.

4.2.1 Occlusion management

Many occlusion management approaches have been proposed Elmqvist and Tsigas, 2008. Multiple viewports show the data from different perspectives Wang Baldonado, Woodruff, and Kuchinsky, 2000. This does not help when the target is strongly occluded from *all* possible viewpoints (R4). Virtual X-ray methods make targets visible by turning occluders (half-)transparent Burns and Finkelstein, 2008.Kruger, Schneider, and Westermann, 2006 proposed ClearView, which interactively focuses on specific areas in the data while preserving context without visual clutter by modulating transparency. Correa and Ma Correa and Ma, 2011 proposed visibility-driven transfer functions (TFs) to maximize the visibility of selected data-intervals. Designing good TFs is still challenging (R2) in general: For instance, in baggage inspection, a dissimulation strategy is to hide a threat among same-density objects, so TF editing cannot easily remove occluders but keep the target (R4) Traoré and Hurter, 2017. De-occluding a tumor from surrounding similar-density tissue in medical scans is a similar situation Preim et al., 2016. Rezk-Salama and Kolb Rezk-Salama and Kolb, 2006 also considered the voxels' occurrence on the cast ray, besides their densities and positions.Hurter et al., 2014a removed occluders in 2D images, volumes, and trail-sets by deforming (pushing them away) in a focus area. Occludes are selected based on data value-ranges, thus we have here the same limitation as ClearView (R4). Li et al., 2012 proposed luggage virtual unpacking where targets are cleared by interactively moving away occluders. This, however, alters the *context* (relative

position, connectivity) where occluders occur (R3). Recently, an interactive volumetric data exploration via direct voxel manipulation was proposed ,Traoré and Hurter, 2017. Extending such approaches in a DVR setting to more complex deformations or changes of the data-in-focus is computationally challenging (R1).

4.2.2 Lenses and deformations

An interactive lens is a lightweight tool to solve localized visualization problems by temporarily altering a selected part of the visual data representation Tominski et al., 2016, and hence provides focus-and-context (F+C) solutions to volumetric data occlusion (R2). Parametrizable lens properties include position, shape, appearance, size, orientation, and selection of included data (focus). The lens *shape* is usually chosen to meet the requirements of the application and is linked to the lens function. Most lenses are circular Tominski et al., 2006 (a design we also choose), rectangular Kincaid, 2010, or adapt their shape to the data-in-focus Pindat et al., 2012; Thiede, Fuchs, and Schumann, 2008. The lens *position* and *size* can be changed manually by the user, or automatically to guide users toward interesting events in the data Tominski, 2011 or along a path of interesting events Alvina et al., 2014. 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 DVR face spatial selection and occlusion challenges. Magic Lens Wang et al., 2005 addresses these by rendering occlusions with lower opacity and magnifying pre-computed features interactively or automatically in a pre-segmented dataset. This, however, does not provide an (interactive) way to deal with similar-density occluders and targets (R4) and does not propose local exploration of the target context (R2). GlyphLens Tong, Li, and Shen, 2017 removes occluding glyphs by pulling them aside through animation. However, this covers only glyph-based and not DVR visualizations.

Lenses can create discontinuities between their inside and outside areas. Deformations can be a solution to this. Hsu, Ma, and Correa, 2011 create flexible deformations by non-linearly sampled rays to smoothly project objects at multiple levels of detail. However, this is computationally expensive and far from real-time (R1). Exploded views are used to partition a volume into several segments Sonnet, Carpendale, and Strothotte, 2004; Bruckner and Groller, 2006. This strongly reduces occlusion but distorts the context (R3) and requires some sort of data pre-segmentation which takes time to compute (R1). Correa, Silver, and Chen, 2007; Correa, Silver, and Chen, 2006 allow one to manipulate the geometry of a data object.McGuffin, Tancau, and Balakrishnan, 2003 performed deformations using peeling to see hidden parts of the data. Such techniques remove potentially important contextual information surrounding the target, which makes it hard to see how the target is embedded in its context (R2).

Deformations can reveal specific data features by using a precomputed segmentation. Tong et al., 2016 proposed a deforming lens which moves streamlines to observe the inner part of streamline bundles. Other techniques performed deformations using surgical metaphors, such us Islam, Silver, and Chen, 2007; Correa, Silver,

Technique	R1	R2	R3	R4	Use-cases
McGuffin, Tancau, and Balakrishnan, 2003	+	–	+	+	1
Sonnet, Carpendale, and Strothotte, 2004	+/-	–	+	+/-	5
Wang et al., 2005	+	–	+	+/-	6
Bruckner and Groller, 2006	+/-	–	–	+	6
Correa, Silver, and Chen, 2006	+	–	+	+	8
Correa, Silver, and Chen, 2007	+	–	+	+	8
Cui et al., 2010	+	–	+	–	5
Hsu, Ma, and Correa, 2011	–	+/-	+	+	4
Hurter et al., 2014a	+	+/-	+	–	6
Wu and Popescu, 2016	+	–	+	–	5

TABLE 4.1: Related work selection *vs* requirements R1...R4 ('+': good, '+/-': average, '–': limited) and the number of use-cases (datasets) used to demonstrate these methods.

and Chen, 2006 to show hidden parts of a volume. Such techniques do not offer tools for local manipulation of the viewpoint that allows seeing a target under multiple perspectives (R2) while keeping the global context.

Using non-straight-line rays is another way to avoid occluders. Cui et al., 2010 propose curved (Bézier) rays to support this. Wu and Popescu, 2016 refine this further to create multi-perspective views. These approaches require a quite careful ray-path planning in advance, so they are not directly aimed at a lens effect (R2). Also, the target should be accessible from the viewpoint via a (possibly curved) path (R4). Also, from the presented examples, although two examples of DVR models are given, it seems these techniques mainly address de-occlusion in large 3D polygonal scenes such as terrain and city models.

Table 4.1 summarizes the main advantages and limitations of a set of volumetric exploration techniques selected from the ones mentioned above which are close to our proposal. As visible, none of the techniques scores high on all requirements. The table also lists the number of different datasets and/or use-cases these techniques were tested on. We will validate our proposal on a similar number of use-cases (section 4.5).

4.2.3 Detailed contributions

Summarizing the above discussion on the requirements and related work on occlusion management, we propose a new technique which combines high-quality DVR with a fast, versatile, and easy to use, lens to support the interactive exploration of occluded data in volumes. In the classification of view deformations by CCarpendale, Cowperthwaite, and Fracchia, 1997, we use a nonlinear radial distortion through an interactive lens to remove occluding items and keep the global context while magnifying a partially occluded item. Related to volumetric lens techniques, we frame our contribution as follows: We propose an interactive deforming lens that magnifies and pushes aside occluding objects located in front of a designated focal point which meets the four requirements; the combination of flexible and real-time interactive changing of the focal point, custom bent rays used for DVR,

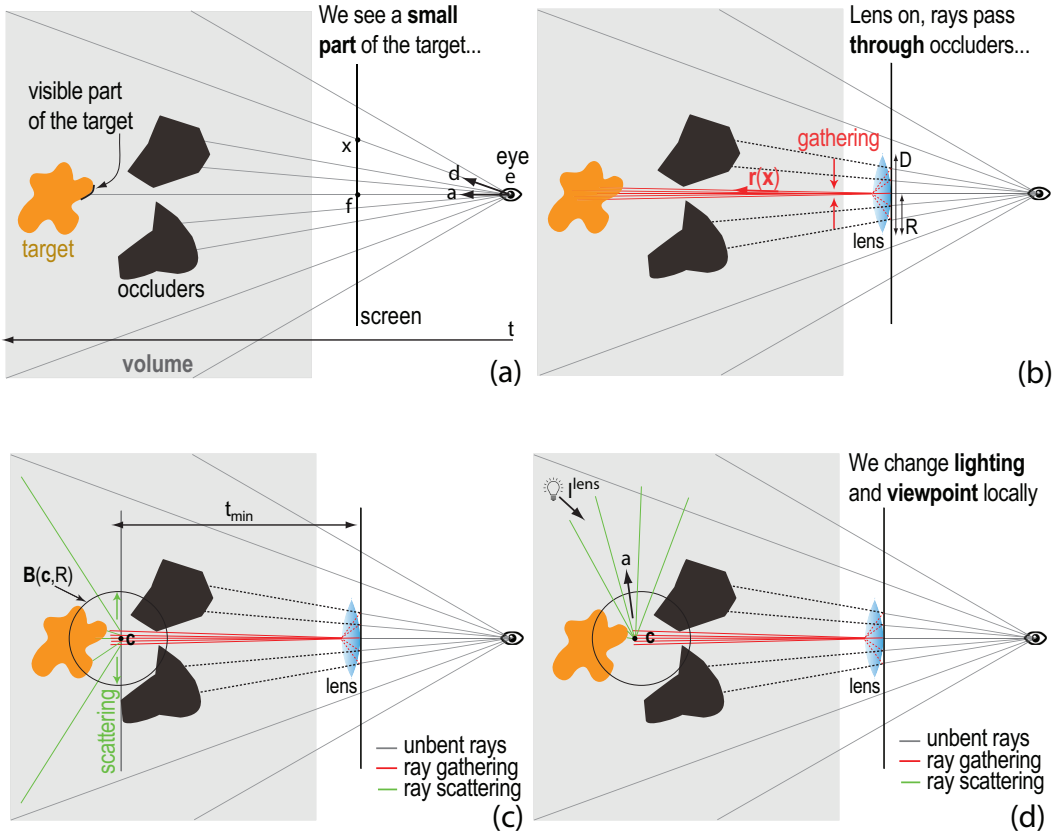


FIGURE 4.1: Obstruction-free lens working. A target is (mostly) hidden by occluders in front of it. (a) Classic DVR shows a small part of the target. (b) Our lens gathers rays to avoid occluders (subsection 4.3.1). Once close to the target, rays follow again their initial paths. Yet, only a small part of the target is visible. (c) Scattering rays makes the full target visible (subsection 4.3.2). Finally, we adjust the local viewing and lighting directions $\mathbf{a}$, $\mathbf{l}^{lens}$ (subsection 4.3.3).

lens deformation, and shading and transfer function in the focus area allow us to provide *on the fly* a range of perspectives of the targets, without having to change the viewpoint or manipulate complex parameters in multiple linked views.

4.3 Principle

Consider the typical DVR algorithm: Given a scalar volume $V \subset \mathbf{R}^3 \rightarrow \mathbf{R}$, each pixel $\mathbf{x} \in I$ in the DVR image $I \subset \mathbf{R}^2$ thereof corresponds to the compositing of sampled data along a ray that passes through V and ends at $\mathbf{x}$. In classical DVR (Figure 4.1-a), such rays are defined by the eye position $\mathbf{e}$ and a ray direction unit vector

$$\mathbf{d} = \frac{(\mathbf{x} - \mathbf{e})}{\|\mathbf{x} - \mathbf{e}\|} \tag{4.1}$$

. Consider now a focus point $\mathbf{f} \in I$ (the lens center) and a lens radius $R > 0$. We modify all rays passing through the lens (focus/0 area $D = \{\mathbf{x} \in I | \|\mathbf{x} - \mathbf{f}\| \leq R\}$ in order to de-occlude, magnify, and emphasize a target object. Our ray behavior

is divided into three steps: (1) Provide a clear view of the target by moving closer to it and by pushing occluders aside. (2) Set a wide field-of-view (fisheye) to better see the target. (3) Modify the parameters of the lens, lighting, and opacity TF in real time to better explore the target. These steps are detailed next.

4.3.1 Creating an unobstructed view

The scenario we address is as follows: Given a volume V, users produce a DVR thereof, using whatever suitable TFs and other parameters are applicable. When examining V from various viewpoints, (at least) one viewpoint $(\mathbf{e}, \mathbf{d})$ is found from which some intriguing structure is *partially* visible in I. We call this structure the *target*. Users next want to quickly and easily unravel the target. For this, we proceed as follows: We first *gather* all rays passing through the lens pixels (focus area D) to follow the lens' axis vector

$$\mathbf{a} = \frac{(\mathbf{f} - \mathbf{e})}{\|\mathbf{f} - \mathbf{e}\|} \tag{4.2}$$

. As explained above, at the location $\mathbf{f}$ of the lens center, we do see an interesting partially occluded target. Hence, by definition, the gathered rays pass *through* occluders to hit this target, otherwise we would not see it. We control gathering by setting the ray direction passing through $\mathbf{x} \in D$ to

$$\mathbf{r}(\mathbf{x}) = (1 - \alpha)\mathbf{a} + \alpha\mathbf{d}, \tag{4.3}$$

with $\alpha \in [0, 1]$. When $\alpha = 0$ (default), all rays follow the lens axis $\mathbf{a}$, thus, can best pass through obstacles. When $\alpha = 1$, rays follow a straight path. Changing α with the mouse wheel smoothly navigates between the lens effect, *i.e.* opening up a 'hole' in the volume to see the target, and classical DVR.

4.3.2 Setting a wide field of view

Once the rays pass obstacles (subsection 4.3.1), we want to *scatter* them so as to best sample the target. Consider that this target is at some depth $t_{target} > 0$ within V. After the rays pass the occluders, but before they hit the target, *i.e.*, travel past a distance $t_{min} < t_{target}$ through V, we deflect (scatter) them so as to best sample the target. For this, we set the parametric position of a ray point to

$$\mathbf{p}(\mathbf{x}, t) = \mathbf{r}(\mathbf{x})t + \beta(\mathbf{x} - \mathbf{f})(t - t_{min}) \tag{4.4}$$

for any pixel $\mathbf{x} \in D$ and any $t \geq t_{min}$. Here, $\beta \geq 0$, adjusted via the mouse scroll wheel while pressing the Shift key (Figure 4.1-c), controls the ray scattering: Small values magnify a small volume area close to the ray $\mathbf{r}(\mathbf{x})$; larger values sample more of the volume behind the lens. Intuitively, this is as if we moved a magnifying lens to a depth t_{min} inside V. Summarizing, after the user finds an interesting but partially occluded target using *standard* DVR, our lens squeezes rays to pass between occluders and next fans them out to explore the target.

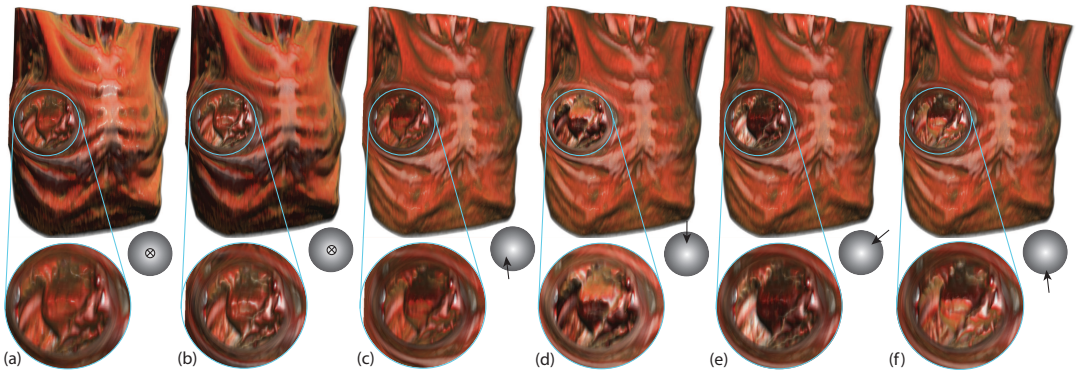


FIGURE 4.2: Changing lighting parameters in the lens. (a) Constant global specular coefficient. (b) Specular coefficient high in the lens and low outside. (c-f) Changing the in-lens light vector yields the effect of a flashlight rotating around the target. The ball icons illustrate the local light vector direction.

4.3.3 Interactive exploration of the target

To achieve a more effective exploration, we can interactively modify several parameters of the DVR and the lens, as follows.

Lens radius: The radius R, telling how big is the 'hole' to open up in the volume to see the target, is set via the mouse wheel. The parameters α and β (affecting the gathering and scattering of rays respectively) are set by the mouse wheel and modifier keys. The value t_{min} (depth from which scattering starts) is set using the arrow keys.

Lens axis: Users can rotate the lens axis $\mathbf{a}$ using a virtual trackball activated by the right mouse button. Changing $\mathbf{a}$ effectively samples the target from many viewpoints, allowing the user to look 'around' it to see parts which are not visible from the current viewpoint, but *without* actually changing the viewpoint. This is of high added value, since changing the viewpoint can bring us to a view where the target is fully invisible, so we do not know where precisely to activate the lens anymore. Figure 4.4 shows three such local rotations for the baggage dataset introduced in Figure 4.5. From these, we see that the star-shaped target is relatively thick.

Lighting: We modify the volumetric Phong lighting parameters to better explore the target, as follows. Let

$$\mathbf{c} = \mathbf{e} + t_{min}\mathbf{a} \tag{4.5}$$

be a point at depth t_{min} along the lens axis, and let $B(\mathbf{c}, R)$ be a sphere of radius R around this point (Figure 4.1b). We call voxels in this sphere 'in focus', and all other voxels in V 'out of focus'. Let ϕ be the specular term coefficient, set to a high value (default: one). First, for all voxels $\mathbf{x} \in B(\mathbf{c}, R)$, we use a specular coefficient

$$\phi(\mathbf{x}) = \phi(1 - d) \tag{4.6}$$

, where

$$d = \frac{\|\mathbf{x} - \mathbf{c}\|}{R} \tag{4.7}$$

. For all voxels outside $B(\mathbf{c}, R)$, we use $\phi(\mathbf{x}) = 0$. Hence, voxels close to the focus point $\mathbf{c}$ appear highly specular; further away from $\mathbf{c}$, voxels get less specular, and voxels out of focus are purely diffuse. Secondly, we allow the user to locally rotate the light vector using the same trackball mechanism as for the lens axis rotation. Let $\mathbf{l}^{lens}$ be this vector, and let $\mathbf{l}^{global}$ be the global light vector used by standard DVR. For all voxels in focus, we use a light vector

$$\mathbf{l}(\mathbf{x}) = (1 - d)\mathbf{l}^{lens} + d\mathbf{l}^{global} \tag{4.8}$$

.

As the user rotates $\mathbf{l}^{lens}$, the light direction will visibly change in the middle of the lens, stay constant outside it, and smoothly change in between.

The above two mechanisms combined yield the effect of a moving flashlight turning around a shiny target embedded in a constantly-lit diffuse scene. Figure 4.2 shows these mechanisms for a chest CT dataset containing a deeply buried tumor (the dataset and use-case are described in detail in subsection 4.5.3). We see how turning the light highlights small-scale details on the target surface (tumor) without changing the viewpoint or lens location. Moreover, the high specularity in the lens attracts the user's attention to this area; the diffuse lighting outside the lens put less emphasis on the context area.

Opacity: We modify the opacity transfer function along a similar idea as for lighting (Figure 4.3). Let $TF_o^{global} : \mathbf{R} \rightarrow [0, 1]$ be the user-chosen opacity function used globally for the volume. Let Γ be a Gaussian pulse of unit height centered at the average density value $\bar{\rho}$ in $B(\mathbf{c}, R)$ and with standard deviation σ. We estimate $\bar{\rho}$ and σ by considering the density ρ at 150 points randomly sampled inside $B(\mathbf{c}, R)$. Higher values for the sample count yield visually very similar results for our tested volumes of up to 500^3 voxels, while requiring (slightly) more computation costs. Then, for voxels in $B(\mathbf{c}, R)$, we use an effective opacity transfer function

$$TF_o = TF_o^{global} + (1 - d)\Gamma \tag{4.9}$$

. For voxels outside B, we use TF_o^{global} (standard DVR). This is useful when the user sets TF_o^{global} to make most out-of-lens voxels relatively transparent. In that case, TF_o will still make voxels in B opaque, thus allowing to see the in-focus structures better. Figure 4.2 has been generated this way.

4.3.4 Smooth transitions

If we bend rays passing through the lens pixels D (Equation 4.3 and Equation 4.4) and trace rays starting at pixels in $I \setminus D$ as straight lines, discontinuities appear at the lens borders. We solve this as follows. Let $\mathbf{p}(\mathbf{x}, t)$ be the voxels along a lens ray starting at screen pixel $\mathbf{x}$, as computed by Equation 4.4. Let $\mathbf{p}^{line}(\mathbf{x}, t)$ be the voxels along a straight-line ray starting at $\mathbf{x}$, *i.e.*, computed using $\alpha = 1$ and $\beta = 0$ in Equation 4.3 and Equation 4.4 respectively. For every value t along every such ray, we compute the interpolated ray

$$\bar{\mathbf{p}}(\mathbf{x}, t) = (1 - f(d))\mathbf{p}(\mathbf{x}, t) + f(d)\mathbf{p}^{line}(\mathbf{x}, t) \tag{4.10}$$

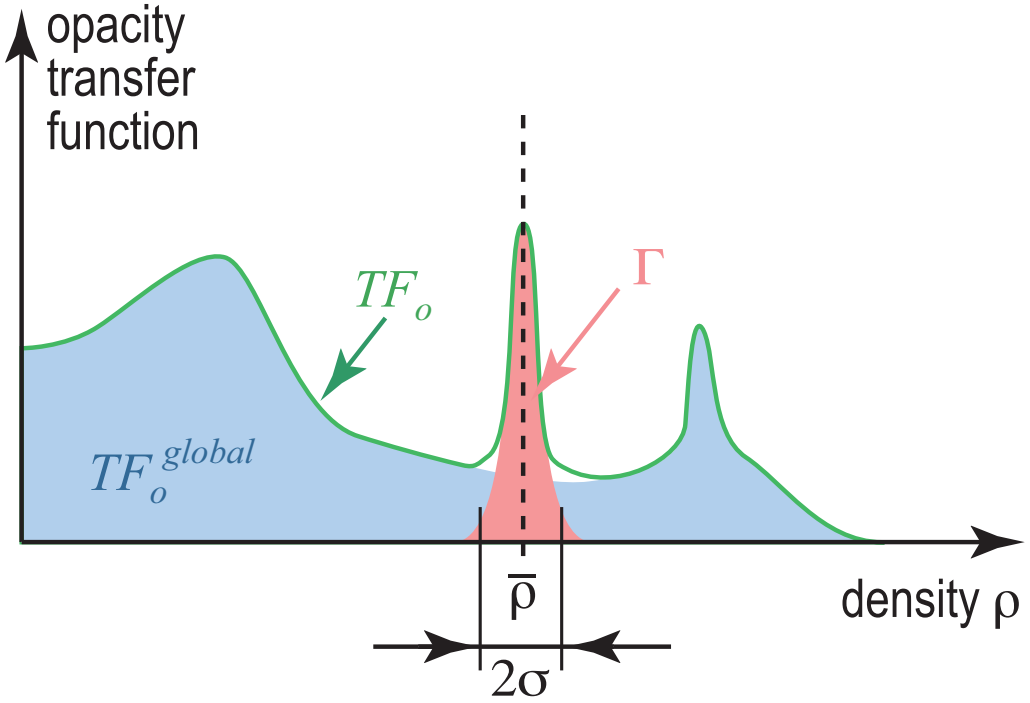


FIGURE 4.3: Construction of local transfer function TF_o. See subsection 4.3.3.

, where d is the distance of $\mathbf{x}$ to the lens axis (normalized to unit by dividing it by R) and $f : [0,1] \rightarrow [0,1]$ is an interpolation function. Next, we use the rays $\bar{\mathbf{p}}(\mathbf{x}, t)$ to compute the DVR by standard composition. This way, rays effectively vary smoothly from their bent versions (close to the lens axis) to straight lines (outside the lens). Setting $f(d) = d^2$ keeps the interpolation transitions close to the lens border, so most of the lens is dedicated to show the desired fish-eye effect.

Separately, we use a slow-in/slow-out animation Dragicevic et al., 2011 to introduce the lens effect. When activating the lens, we vary α and β from their defaults ($\alpha = 1$, $\beta = 0$, *i.e.* straight-line classical DVR) to their actual user-set values, compute the volume rendering on-the-fly, and display the resulting images. The effect resembles gradually opening a hole in the volume – see the associated video. The speed increase at the start of the animation helps one to quickly see what is revealed in the lens; the decreasing speed at the end helps seeing where the pushed-away occluders actually go. This also gives some semantic to the moving shapes, allowing one to interpret the motion as a magnification of a target, and to keep the focus on visual entities during this transition. When deactivating the lens, we play back the animation in the opposite sense, which suggests closing the opened hole in the volume.

4.4 Implementation

We implemented our occlusion-free lens by modifying a standard DVR ray caster, publicly available in NVIDIA CUDA's SDK **cudasdk**. We modified this ray caster to

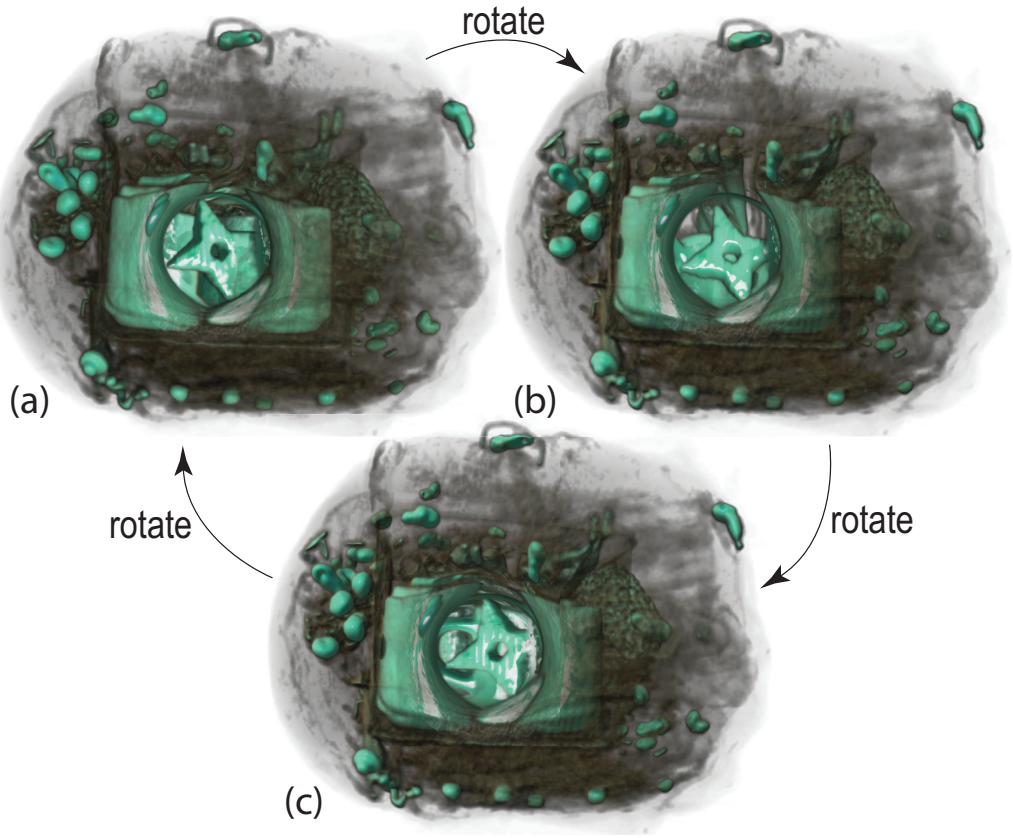


FIGURE 4.4: Performing local rotations in the lens allows better seeing the shape and thickness of the partially occluded target object (ninja star).

incorporate the new ray definition (Equation 4.3 and Equation 4.4), the lens effect, and the local per-voxel Phong lighting parameters, all controlled via keyboard and mouse. On a PC with 16 GB RAM and a GeForce GTX TITAN X card, we achieve 15 frames per second for volumes up to 512^3 voxels at a 1900×1200 pixels screen resolution. All in all, adding our lens to an existing ray caster should pose no significant implementation problems.

4.5 Application Scenarios

We next demonstrate our obstruction-free lens via five use-cases considering scalar density volumes from baggage inspection, 3D flow simulation, radiology, air traffic planning, and diffusion tensor imaging.

4.5.1 Baggage inspection: An unusual blunt object

In airports, security agents deal with volumetric data exploration during baggage inspections. While automatic systems can detect densities of harmful substances such as C-4, TNT, and nitroglycerin, or prohibited articles (threats) like classical firearms and knives, unusual threats are hard to find. Four main concealment strategies exist Traoré and Hurter, 2017:

- **Superposition**: A threat may be sheltered among dense materials. While possible to see through such a 'shield' using high penetration (enhanced X-ray power) or image processing (contrast improvement), such techniques are not universally available and also require fine-tuning many parameters, which slows down inspection.
- **Location**: Objects located in the corners, edges, or in the luggage's frame are very hard to spot.
- **Dissociation**: One can conceal a threat by spreading its parts in the luggage, *e.g*, by disassembling a weapon and scattering its parts.
- **Lure**: A minor threat (lure) like small scissors is clearly visible and catch the security agent's attention who can miss the real threat.

Baggage labeled as suspicious by human inspection or automated scan heuristics must be checked by human agents. Besides time-consuming physical unpacking, one can use 'virtual unpacking' tools that segment the 3D scan by a density-based confidence measure and next move the segmented objects away by animation to reduce occlusion Li et al., 2012. Such systems have been patented and used in production **patent**. However, when the automatic segmentation is not optimal, the user must manually change its parameters, repeat the segmentation and animation, which goes back to being time-consuming.

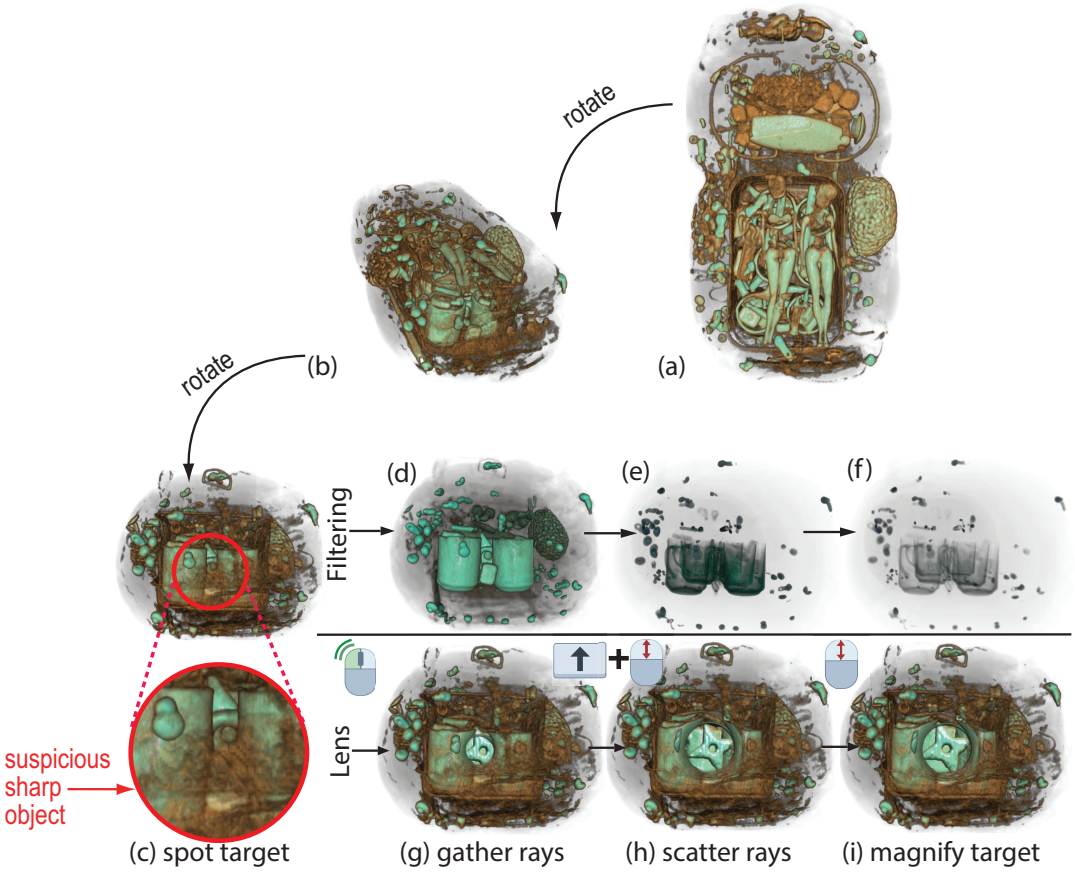


FIGURE 4.5: (a-c) A baggage scan is viewed from different angles. In view (c), a suspicious sharp object is spotted between a set of mugs. (d-f) Filtering densities using a classical 1D opacity transfer function removes progressively more of the occluders (mugs), but also the target. (g) The user applies the lens on the target object (double-click). An animation starts opening the lens, rays are gathered to pass through occluders. Halfway the animation, the object is magnified, but only the area close to the lens is visible. (h) The fish-eye field of view at the end of the animation scatters rays to fully show the target. (i) The lens is increased to magnify the target (mouse scroll).

Consider the baggage scan in Figure 4.5 ($283 \times 189 \times 344$ voxels, dataset obtained from an actual airport scan). Automatic baggage inspection systems will not detect anything suspect here. However, while visually exploring this baggage from different angles (Figure 4.5a-c), we see an object hidden between a set of mugs. To reduce occlusion, a common solution in baggage inspection is to filter materials by density in order to show or hide subsets of the volume. However, for our dataset, the suspect target has almost the same density as the surrounding mugs, so removing the latter also removes the target (Figure 4.5d-f). Using the obstruction-free fish-eye lens helps here: Clicking on the sharp detail visible in Figure 4.5c first gathers rays so they pass through the low-density zone between the mugs (Figure 4.5f). The animation that opens the lens (Figure 4.5e-g) reveals an unobstructed view of the target. However, this shows only a small part of the target. Scattering rays next fully reveals the target (Figure 4.5h). Adjusting the lens size shows a more detailed view of the target (Figure 4.5i). Next, locally turning the viewpoint around the target (Figure 4.4) allows the agent to decide that the target is a shuriken (Japanese ninja star weapon). Since the object is very thick and blunt (see Figure 4.4), it is not an actual weapon, thus not a threat.

We evaluated our lens for this use-case by a user study. This evaluation is described bellow in subsubsection 4.5.1.1.

4.5.1.1 Early results

Eight airport security specialists were recruited (ages 23 to 43; experience in baggage scanning 8 months to 20 years; average familiarity with 3D tools, none considering him/herself an expert.

All attended a 20-minute global demo of the lens operation. Next, they were given each a personal training session for using the tool (5 minutes), in which they were instructed on the mouse and keyboard controls. After this, they were asked to work in pairs to examine the above-mentioned baggage CT dataset to form a decision on the nature of the ninja star possible threat (20 minutes of tool usage per person, after which the pair was changed). The idea behind this is that one person operates the tool while the other poses questions or suggest explorations, much like typical airport security operators work with a scanner. In the end, they all separately filled in a web questionnaire covering several questions and also provided open feedback (questionnaire available in the Appendix A).

subsubsection 4.5.1.1 shows the answers. The first question-set (S1) regarded how easy-to-use, generally effective, and effective *vs* other known tools our lens is for *untargeted* inspection, *i.e.*, when no suspect target is partially visible. Here and next, other tools denote classical 2D X-ray or 3D CT scans used in baggage scanning that the subjects know. As Figure 4.8, Figure 4.9, and Figure 4.10 show, the answers (on a 5-point Likert scale) were predominantly positive: The tool is easy to use, is useful, and is actually more useful than known tools for untargeted exploration. The second question-set (S2) regarded how good our tool is to examine *specific* targets which are partially visible. Here again, the answers were predominantly positive

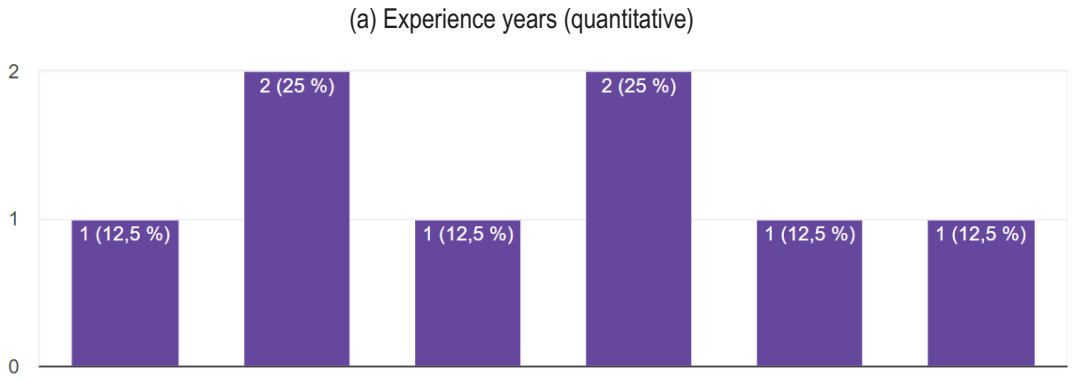


FIGURE 4.6: Evaluation of lens-based baggage inspection (subsection 4.5.1): Experience years.

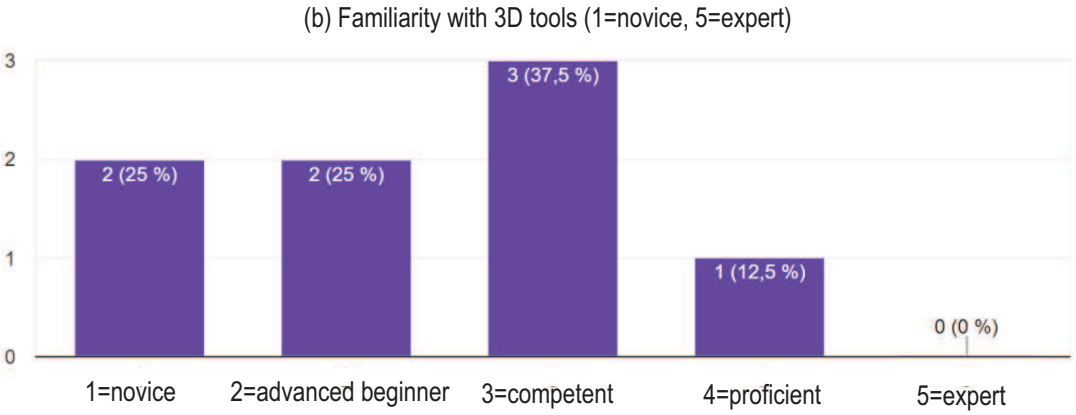


FIGURE 4.7: Evaluation of lens-based baggage inspection (subsection 4.5.1): Familiarity with 3D tools.

(Figure 4.11, Figure 4.12, and Figure 4.13). The main appreciated features of our tool are listed in Figure 4.14.

We next summarize the received open feedback. According to the subjects, our tool can provide them a better perception of the items inside the baggage as compared to the classical 2D single-viewpoint X-ray machinery they routinely use. Quotes from the open feedback: "clear added-value compared to all systems I know"; "this tool is a real gain for examining luggage with uniform and/or high densities"; "definitely better than known tools for examining threats I am not familiar with / I have not seen before". However, our tool should not be used for the typical carry-on baggage inspection which has a very small allowed inspection time (15 to 20 seconds). Our tool is much more interesting for inspecting checked-in baggage, where inspection time-windows are up to 3 minutes. The perceived added value for this use-case is also higher: Opening up checked-in baggage for manual inspection is much more complicated and time-consuming than for carry-on baggage. Moreover, the only system for inspecting checked-in baggage that the subjects knew of is a scanner that aims to *automatically* detect threats via X-ray imagery; this system suffers from false positives, so a manual examination tool like ours could quickly eliminate such false positives, and thus the delays of opening up checked-in baggage. Finally, several subjects suggested that adding a function to display a classical 2D slice view (activated by a key press and aligned with the focal point) would be useful since this

(c) **S1**: Ease of untargeted exploration (1=very poor, 5=very good)

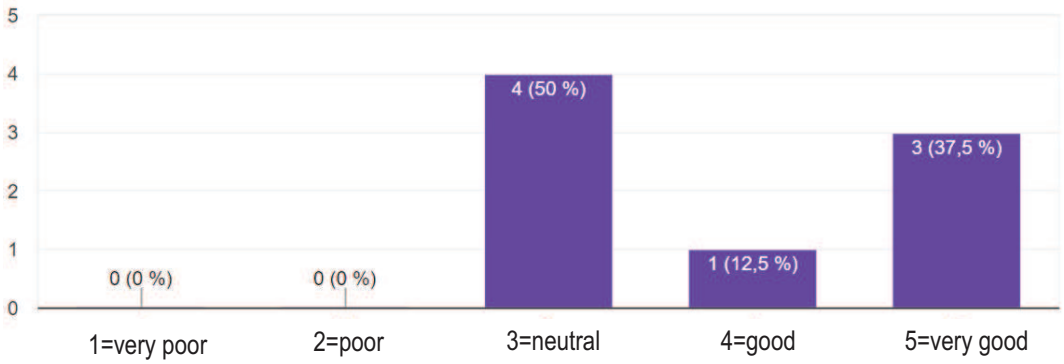


FIGURE 4.8: Evaluation of lens-based baggage inspection (subsection 4.5.1): Scenario 1 - Ease of untargeted exploration.

(d) **S1**: Tool's value for untargeted exploration (1=very poor, 5=very good)

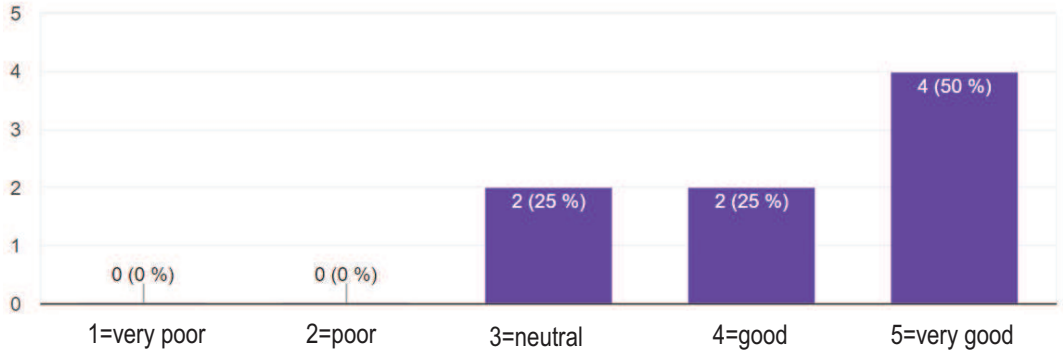


FIGURE 4.9: Evaluation of lens-based baggage inspection (subsection 4.5.1): Scenario 1 - Tool's value for untargeted exploration.

(e) **S1**: Tool's value, untargered exploration, *vs* other used tools (1=very poor, 5=very good)

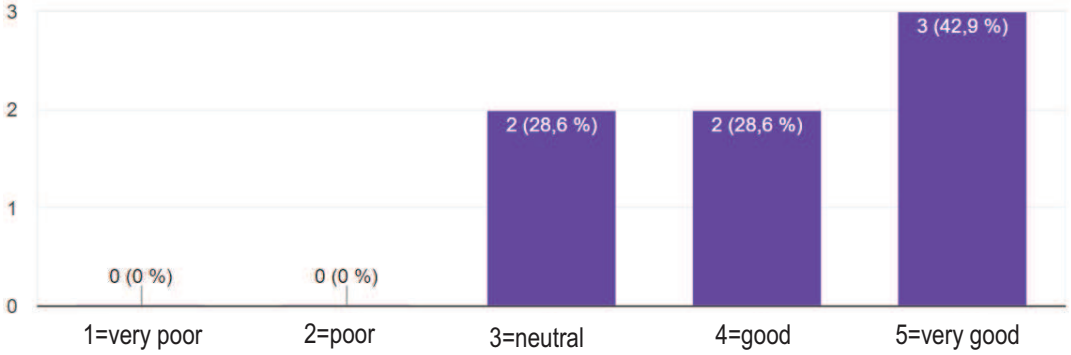


FIGURE 4.10: Evaluation of lens-based baggage inspection (subsection 4.5.1): Scenario 1 - Tool's value, untargeted exploration, vs other used tools.

(f) **S2**: Ease of targeted exploration (1=very poor, 5=very good)

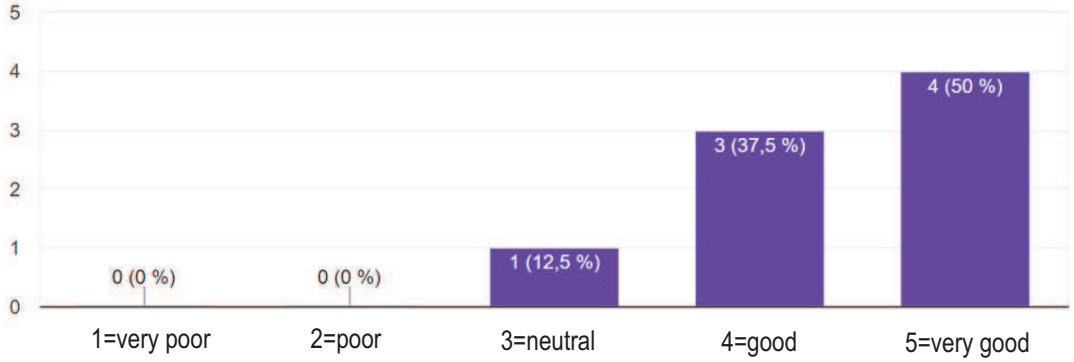


FIGURE 4.11: Evaluation of lens-based baggage inspection (subsection 4.5.1) : Scenario 2 - Ease of untargeted exploration.

(g) **S2**: Tool's value for targeted exploration (1=very poor, 5=very good)

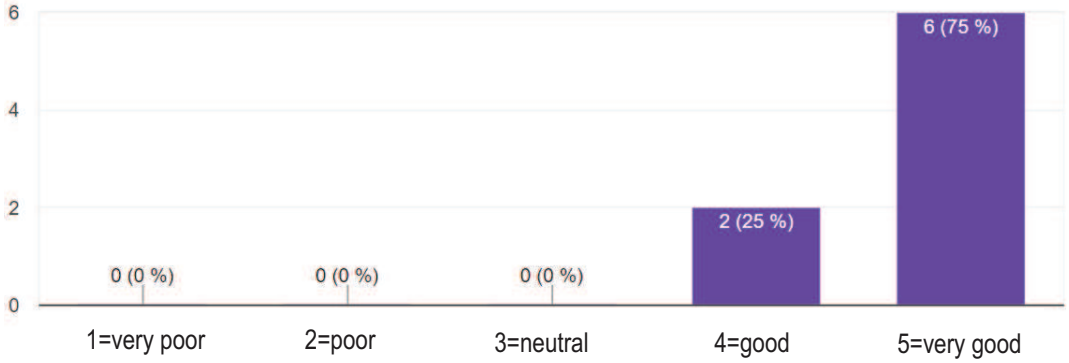


FIGURE 4.12: Evaluation of lens-based baggage inspection (subsection 4.5.1): Scenario 2 - Tool's value for untargeted exploration.

(h) **S2**: Tool's value, targeted exploration, *vs* other used tools (1=very poor, 5=very good)

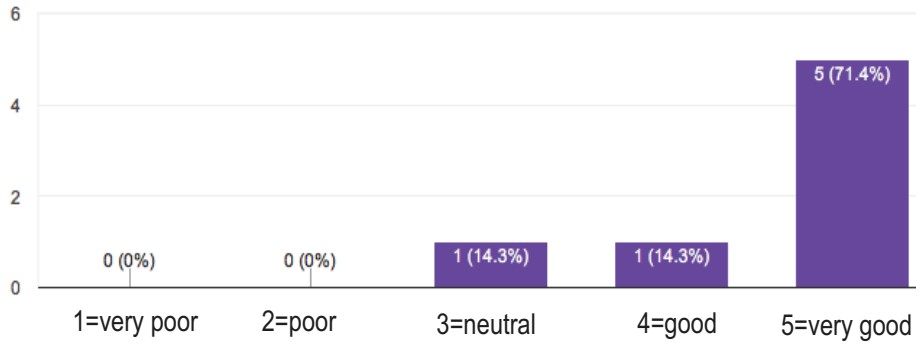


FIGURE 4.13: Evaluation of lens-based baggage inspection (subsection 4.5.1): Scenario 2 - Tool's value, untargeted exploration, vs other used tools.

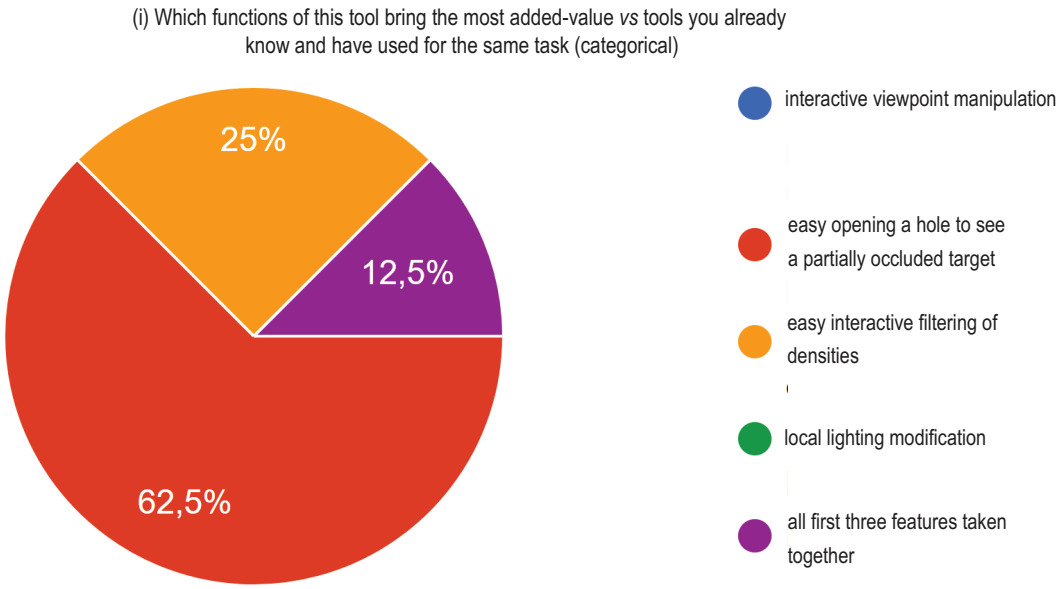


FIGURE 4.14: Evaluation of lens-based baggage inspection (subsection 4.5.1): Which functions of this tool bring the most added-value vs tool you already know and have used for the same task (categorical).

would show additional detail.

4.5.2 Fluid flow: A deep-buried spherical vortex

Flow visualization using streamlines has a long history Brambilla et al., 2012; Merzkirch, 2012. For 3D datasets, a key challenge is to balance the streamline density. Low values allow seeing inner regions in the data but can subsample (miss) patterns. High values show more data but create too much occlusion. We next show how our lens can be used to alleviate problems in the latter case. The dataset Griebel et al., 2004 captures the simulation of water flow in a basin computed on a grid of $128 \times 85 \times 42$ cells using 4595 streamlines with 183K sample points traced by pseudo-random seeding. We convert this set of 3D curves (polylines) to a scalar volume by using GPU-accelerated kernel density estimation (KDE) Lhuillier, Hurter, and Telea, 2017. Similar techniques have been used to compute density maps of 2D trail-sets **cubu**; Hurter, Ersoy, and Telea, 2012; Hurter, 2015.

We first explore the density volume (500^3 voxels) using standard DVR (Figure 4.15). Note that, given KDE's smoothing effect, streamlines appear as finite-thickness tubes rather than pixel-thin curves. After turning the viewpoint a bit, we notice a dense spherical item deep in the data (Figure 4.15a). To see its shape better, we increase opacity; however, this immediately increases occlusion so the item becomes invisible. Conversely, decreasing opacity to reduce occlusion makes the item almost transparent. Our lens solves the problem: In the initial view (Figure 4.15a), we point at the target and turn on the lens. This pushes away the occluding stream bundles, and shows that our item is a set of densely-packed, low-speed, tightly-turning streamlines that create a ball-like vortex (Figure 4.15b). To make sure our target is spherical,

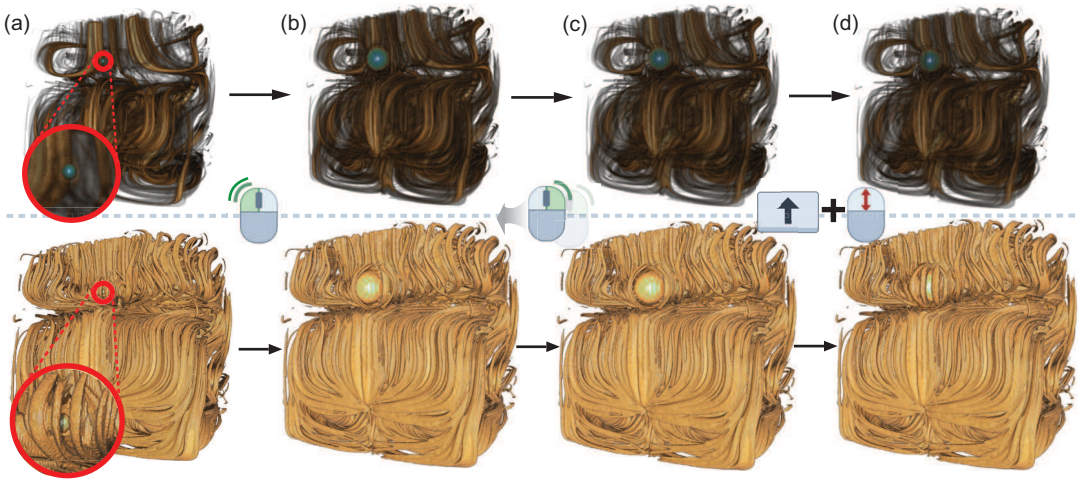


FIGURE 4.15: Flow volume exploration with two different opacity transfer functions (top and bottom rows). In viewpoint (a), we notice a small high-density spherical item. (b) We apply the lens at that location (double click). (c) The directions of rays in the lens are changed to see the whole target in the lens (right click + mouse drag change direction). (d) The lens is gradually closed while keeping the focus area magnified (shift + scroll).

we view it in the lens from different directions, by interactively changing the ray directions in the lens (Figure 4.15c). Finally, we can close the lens but keep the target magnified (Figure 4.15d). Finding the details of this vortex cannot be done using standard DVR. Interestingly, this vortex has also not been discovered by any of the visualization techniques that used this dataset (according to our knowledge) Telea and Wijk, 1999; Griebel et al., 2004; Everts et al., 2009; Lhuillier, Hurter, and Telea, 2017.

4.5.3 Chest scan: A hard to see tumor

In our third use-case, we consider a contrast chest CT scan ($512 \times 512 \times 110$ voxels) of an elderly patient with a sizeable lung tumor. The tumor was detected in a CT scan performed after the patient reported acute chest pain. Typical examination of these scans by the pulmonologist and radiologist in charge involves slice-based views. Figure 4.16a-c and Figure 4.16d-f show two such slice sets (axial, coronal, and sagittal views), produced using typical lung, respectively mediastinal, contrast presets.

Although the tumor is visible in all these views, its exact shape, morphology, and connection to the lung walls are hard to assess. Finding such details on the tumor is essential, explained the doctors in charge, to determine the TNM score **brierley** and also planning treatment. Using standard DVR makes the tumor and its 3D position partially visible (Figure 4.16f). Yet, occlusion from the rib cage and other tissues is still present. Using both TF presets and manually changing the TFs in the 3D Slicer tool **slicer** used to create the DVR could not help de-occluding the tumor without making it partly transparent.

The slice images in Figure 4.16a-f confirm this by showing that the gray values for the tumor and surrounding skin-and-muscle tissue are very similar. This is due to the fact that the tumor had grown rapidly and started necrotizing, which filled it with fluids, making its density very similar to that of the obstructing (skin and muscle) tissue, explained the pulmonologist. Hence, one cannot remove such occluding tissue in a classical DVR setting by opacity TF manipulation without also removing the tumor. This makes examining this specific tumor harder than for regular cases.

We next used our lens to examine the tumor. Figure 4.2 shows several sample snapshots. We see that the tumor is significantly more visible when using the lens than when using standard DVR (Figure 4.16d), both in terms of removing the occluding tissue and in terms of the tumor's opacity – compare the inset in Figure 4.16d with the images in Figure 4.2. Secondly, relighting the tumor from various directions allows one to see small-scale morphological details such as the tumor's surface shape and its connection via protuberances and veins with the lung walls.

We asked the two medical specialists (pulmonologist and radiologist) in charge to state the potential advantages and/or limitations of our lens as compared to standard slicing and DVR techniques, after a 20-minute usage of the tool. Both specialists have over 10 years of medical experience in treating lung cancer, and routinely use several slicing and DVR tools. They work in a private hospital in Belgium and are not actively associated with medical imaging research. Our identities were hidden from them during the lens evaluation. The provided input can be summarized as follows: The occlusion-free lens is definitely easier and faster to use than classical DVR and/or slicing techniques. It is especially more effective than these to get a quick, first impression of a deep buried anatomical detail. Changing the lens' parameters by direct interaction is as simple as changing window/level functions in a typical slice-based tool, and is definitely simpler than tuning typical DVR parameters to obtain similar results. This 'entices' the user to explore, which is a good aspect. The fact that the lens minimizes viewpoint change (volume rotation), *i.e.*, after a suitable viewpoint was found from which a (small) part of the target is visible, one doesn't need to change this viewpoint, is a strong feature, as 3D viewpoint changes are disruptive and cost time. This is important in a cost-aware environment where specialists have very limited time (about 20 minutes) to assess a CT scan. However, the lens should not *replace* classical slice-based exploration, which shows small-scale details better. In the context of the current dataset (Figure 4.16), the lens was useful to both confirm the TNM score (T3 grade tumor, 6.5 cm in size) found via the 2D slices, but much more so for understanding how and where the tumor is connected to surrounding tissue, which is very hard to do using only 2D slices.

4.5.4 Aircraft trajectories: Outliers in the French sky

We next consider a task from air traffic planning – detecting and studying outliers in large-scale datasets containing tens of thousands of 3D (latitude, longitude, height) trails of aircraft over a given spatio-temporal region Hurter et al., 2014b. Such datasets are typically viewed using 2D (latitude, longitude) plots where opacity encodes the spatial density of flights – see Figure 4.17a, which shows one day of recorded aircraft trajectories over the French airspace. Figure 4.17(b) shows a detail

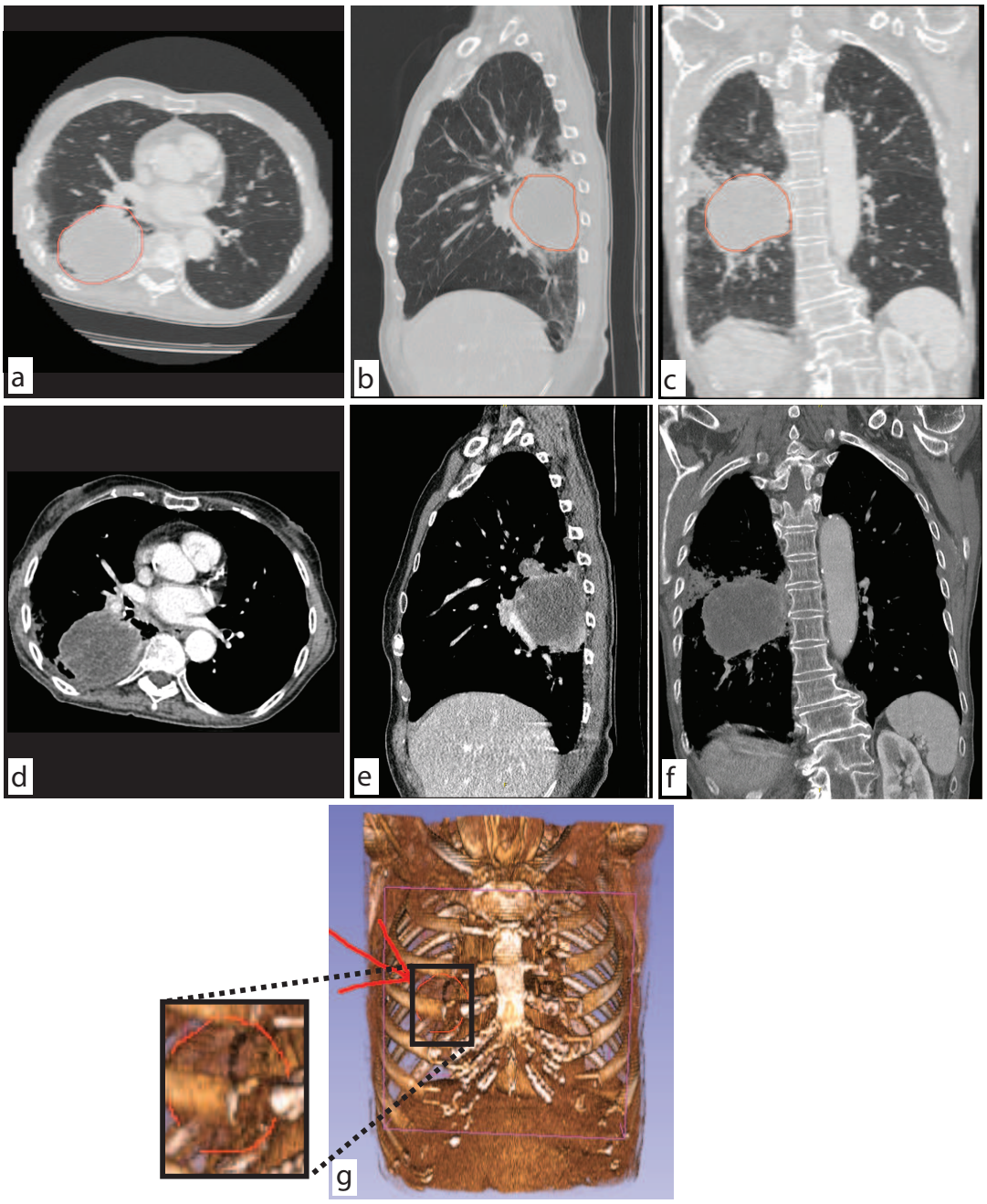


FIGURE 4.16: Lung tumor visualization using slices (a-c) and standard DVR (d). Annotations are manually added by the examiner to delineate the tumor location. Images constructed using the 3D Slicer tool **slicer**.

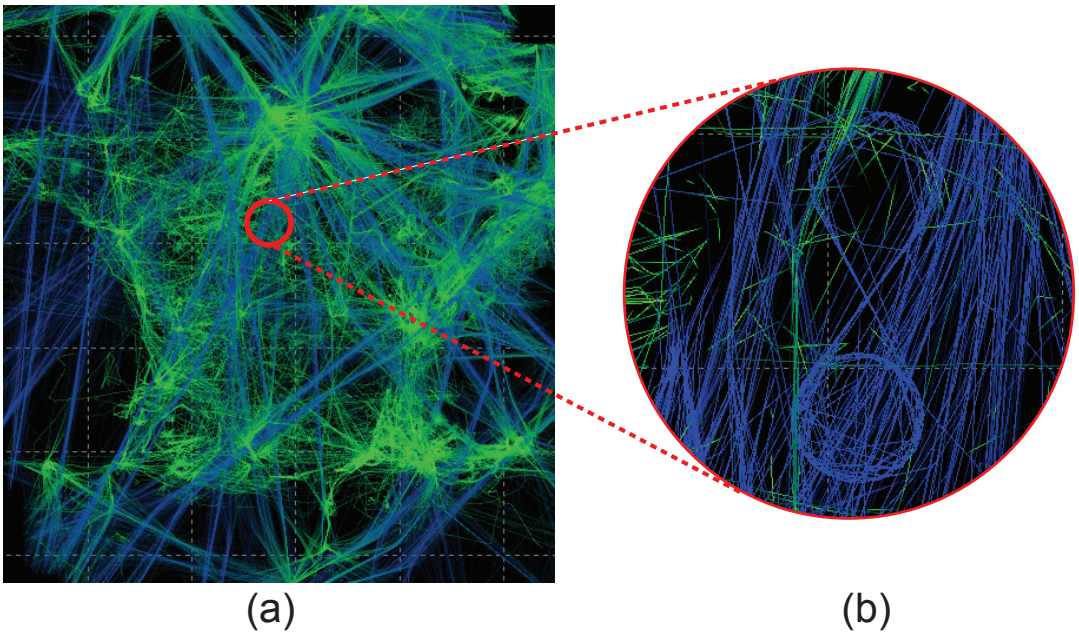


FIGURE 4.17: Visualizing one day of aircraft trajectories over France Hurter, Tissoires, and Conversy, 2009. (a) Overview of all trails. (b) Zoom, filtering, and color mapping techniques are used to highlight an outlier trajectory of an aircraft performing an eight-shaped loop. Revealing this outlier costs significant user effort.

zoom-in, where we can see an abnormal – that is, far from straight or slightly curved – aircraft trail: A tanker aircraft performed an eight-shaped loop as it was waiting to refuel other aircraft. Revealing such patterns using 2D techniques, *e.g.* Hurter, Tissoires, and Conversy, 2009, is very hard. In particular, it is hard to de-occlude these patterns from the overall context of criss-crossing aircraft trails, even when one knows their 2D spatial location.

Our lens can help with this task, as follows. We first convert the set of 3D trails to a 500^3 density volume, using KDE as for the streamline use-case (subsection 4.5.2). Examining this volume via standard DVR shows an outlier trail at some point in space, see curved patterns in Figure 4.18a. Activating the lens on this area and interactively tuning the target depth t_{min} (since we don't know the trail's height) beings the outlier trajectory in focus and pushes away occluding trails (Figure 4.18a). Like in the other examples presented so far, we can quickly change the magnification factor and view direction to better study this trail in context (Figure 4.18b-d). From these images, we easily see that the outlier trail has an eight shape. Revealing this outlier trail using standard 2D visualization techniques Hurter, Tissoires, and Conversy, 2009 costs several minutes. Doing the same using our lens costs under one minute. Also, comparing Figure 4.17b and Figure 4.18b-d, we argue that the eight-shape of the outlier trail is much more prominent, and thus recognizable, in the latter images (made using our lens) than in the former ones. Last but not least, the 3D DVR approach that our lens enhances explicitly encodes flight height information, so our lens can use it by interactively tuning the depth value t_{min} where the lens is focused. This cannot be done with 2D techniques which ignore the depth dimension.

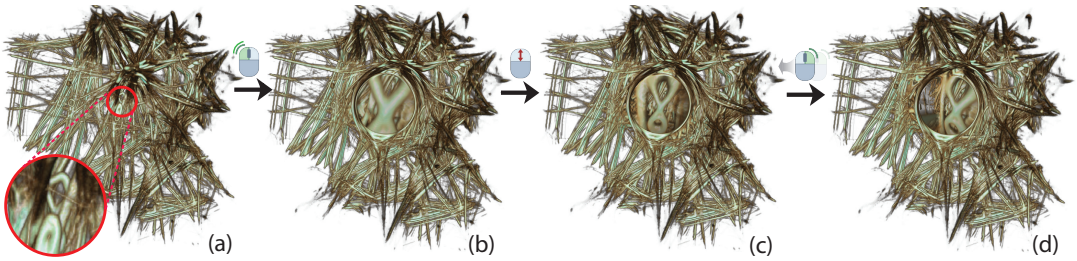


FIGURE 4.18: Inspecting an abnormal aircraft trail. (a) The abnormal trail is spotted in an all-trails view as it is highly curved while all other trails are relatively straight. Activating the lens at the outlier location (b) and changing the magnification factor (c) reveals the trail's eight-shape. (d) Rotating the viewpoint provides spatial insight on the embedding of the outlier in the surrounding trails.

We validated our findings with an air traffic data scientist with more than 10-year experience in air traffic control and planning. She confirmed that this specific eight-shape trail in Figure 4.17(b) is an actual aircraft which performed waiting loops and acted as a fuel supplier for military aircraft. Other comments included the following: Compared to standard 2D visualization techniques, our tool makes detecting outliers easy since there is no need for complex manipulation to reveal such outlier trails. Also, the user does not have to deal with color and alpha mapping parameter-tuning to make specific outliers emerge. Separately, trail visualization easily creates many occlusions leading to either fully opaque areas or too much local overlap, which both hinder seeing and examining specific trails. Our lens does help such cases by distorting the space to locally remove such occlusions. All in all, in the studied dataset (Figure 4.17), the lens was specifically useful since, for high transparency, one would not detect the outlier trail, while for low transparency, one would get a hint of the outlier's existence, but not see it in detail due to too much occlusion; the lens allows using low transparency, but removes the clutter caused by it to reveal the outlier.

4.5.5 Brain fibers: Uncluttering the bridge

Our last use-case considers the exploration of fiber tracts visualized as streamlines of the major eigenvector of a diffusion tensor imaging (DTI) field. Such datasets have a spatially complex structure which makes them hard to explore **assaf08**. In particular, fiber tracts are spread volumetrically over the entire extent of the brain, and create tangled patterns inside which it is hard to see much. DVR techniques are often used to render such tracts, one of the advantages being that close fibers get visually 'merged' to reveal spatially coherent structures, an effect which is not possible when fibers are rendered as polylines. However, DVR methods also create more occlusion, thus difficulties in seeing structures deep within the volume.

We consider an $128 \times 128 \times 51$ DTI volume (same dataset as in **everts15**). We traced 150352 fibers seeded in, and going over, regions of high fractional anisotropy in this volume. We filtered out fibers shorter than 2mm, yielding a total of 120593 fibers to display (6.4M sample points). Next, we converted this fiber-set to a 512^3 density volume, using KDE with a 3D isotropic kernel of radius 15 voxels, like for

the streamline use-case (subsection 4.5.2). Figure 4.19a shows the result, rendered with DVR, with opacity function mapping the fiber density. While terminal fibers are well visible, we cannot see anything inside the volume. Activating the lens in the middle of the volume opens a hole through which a small part of the *corpus callosum*, the fiber bundle wrapping the bridge that connects the two hemispheres, becomes visible. By slightly decreasing opacity (Figure 4.19c), the *corpus callosum* gets clearly visible, appearing as a compact structure, due to the KDE blending of neighbor fibers. Obtaining such a view of the *corpus callosum* only using DVR would be very hard, since transfer functions would either render separated (non-merged) fibers, or else make the fibers surrounding the structure of interest too thick and occluding.

This scenario has the main difference compared to all previous ones. In all earlier cases, the standard DVR of the data (that is, without the lens) showed us a partial small cue of the structure of interest within the volume, and we used the screen-space location of this structure as the focus point where to activate the lens. In this last scenario, there is no point in the original DVR image (Figure 4.19a) from which the *corpus callosum* is even partially visible, due to the high opacity given by the used transfer function. Hence, the user can activate the lens at *any* desired point to peek inside, and towards the center of, the volume. Given the nature of the data, the structure of interest is quite easily visible from most such viewpoints (see lens inset in Figure 4.19b). Once its presence is revealed, the user can next adjust the viewpoint and/or the opacity transfer function to get an optimal view on the target, such as the one shown in Figure 4.19c. Summarizing, we can use our lens also in cases when no partial view of a target is available.

4.6 Discussion

Several points of our lens proposal are worth discussing, as follows.

Lens activation: Our lens can support two types of explorations. First, when the user perceives a *part* of a target of interest in a classical DVR image, the lens can be used to reveal the target in full detail. This *directed* exploration supports the task 'show me more information about *this* item'. The use-cases in subsection 4.5.1-subsection 4.5.4 are of this type. Secondly, the user can open up a DVR volume at a 2D location from which no partial detail is visible. This is useful when we know that there *is* an interesting target buried in the volume even without seeing it (*corpus callosum* use-case in subsection 4.5.5), thus supports the task 'show me the data I *know* it is somewhere in there', or for free exploration to find unknown patterns in a volume, *i.e.* for the task 'show me what this volume *may* hide in it'. In the first exploration type (target not fully occluded), our lens is simple and rapid to use – point, click, and optionally rotate light or viewpoint. In the second exploration type (target fully occluded or not even sure whether an interesting target exists in the data), the lens is equally simple to use, but several tries to select a suitable focus point and lens depth are needed.

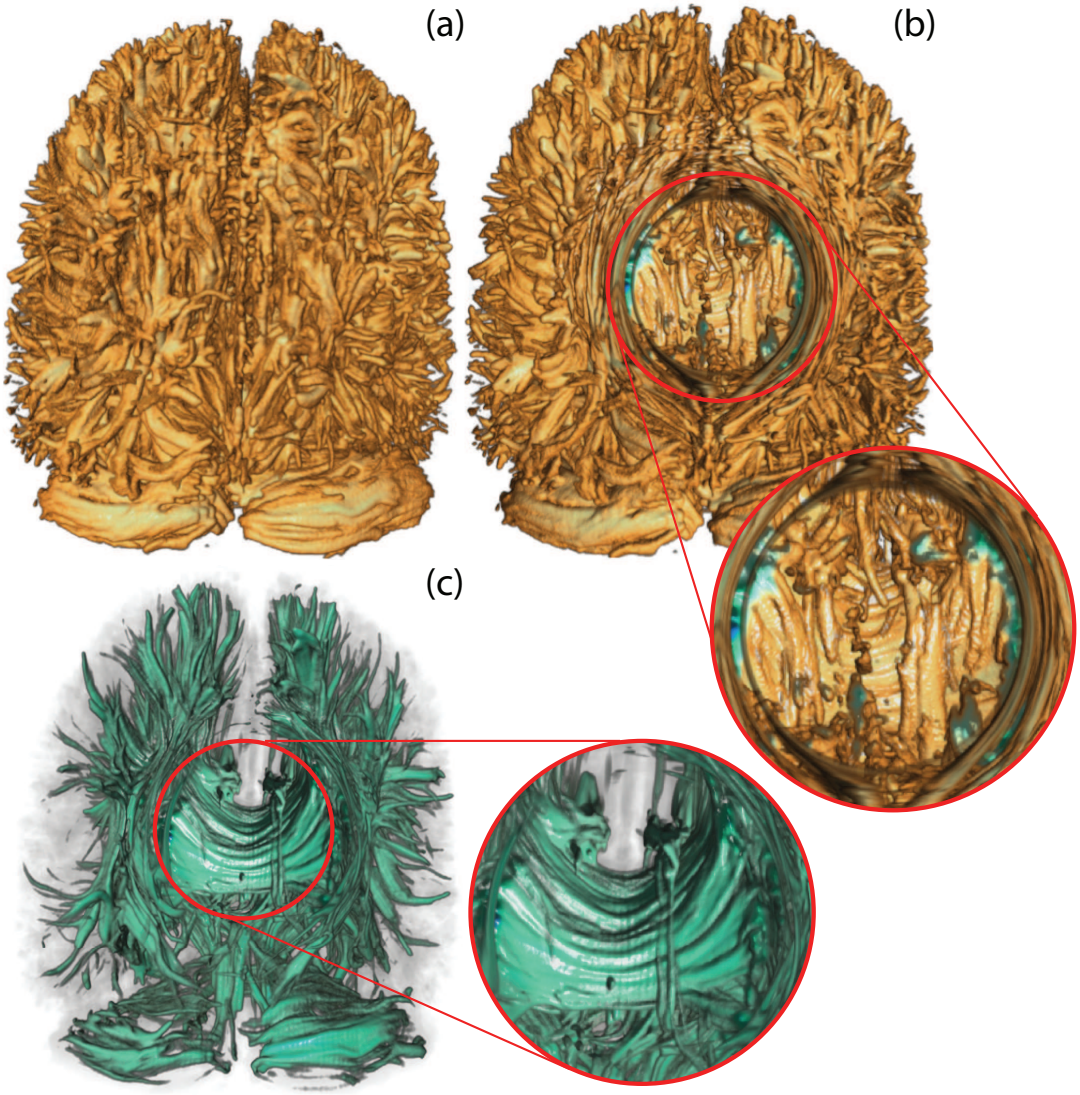


FIGURE 4.19: Revealing the *corpus callosum* in a DVR of a set of DTI tracts.

Lens shape: Occluders are pushed away, and deformed, isotropically (subsection 4.3.2, subsection 4.3.4). This simple lens model requires a single parameter, the lens radius R, which makes its usage easy. The deformations evolve smoothly from the lens center (maximal) to outside the lens (no deformation), see subsection 4.3.4, which effectively blends the local (in lens) focus with the global (out of lens) context (R3). However, this strongly compresses the deformed occluders close to the lens border, making them hardly visible when the lens is fully active. A possible refinement would be to reduce the deformation of the pushed-away occluders while still pushing them away, thereby improving the F+C effect (R3). However, this would occlude areas outside the lens, basically moving occlusion from *inside* the lens to *outside* and close to it. Finding an optimal balance between minimal deformation (so one can recognize the pushed-away occluders) and minimal clutter (so these occluders do not destroy the lens context) is a topic for future work. Separately, deformed rays may intersect with straight rays, thereby sampling the same voxel(s) to different image pixels. We did not observe in our usage any artifacts that can be ascribed to this issue, nor did the other users of our tool. This can be explained by the fact that such ray intersections are relatively few and we use a compositing transfer function, akin to a low-pass filter.

Parameter setting: Our lens depends on several parameters: the 2D lens center $\mathbf{f}$, lens radius R, lens axis direction $\mathbf{a}$, local light direction $\mathbf{l}^{lens}$, scattering start-distance t_{min}, and gathering and scattering parameters α and β. All these parameters are controlled via a mouse-driven virtual trackball, key modifiers, and the arrow keys (section 4.3). As the lens works at 15 frames per second, the user can quickly tune the parameters and see their effect (R1). Moreover, all parameters start with good preset values (section 4.3). A possible refinement would be to pre-segment the target, based on user-given values for $\mathbf{f}$, R, and t_{min}, thereby determining β automatically. However, we believe that manual control of the scattering β is important to allow users to choose their most suitable field-of-view angle. In fact, this flexibility allows a better exploration of the local context (R2).

Implementation: We implement our lens by modifying the ray trajectories constructed in the inner loop (per-pixel raycasting) of a public DVR raycaster **cudasdk**. Apart from this, we change the per-voxel lighting and transfer function based on the voxel location in the lens and the parameters given by user interaction (subsection 4.3.3). Such changes are limited and easily applicable to any (parallel) raycaster.

Limitations: As explained, de-occluding a target requires either a small fragment thereof to be visible (if so, de-occlusion is very simple and fast), or requires the user to choose the lens focus and target depth based on other insights (which, as explained, requires more trial-and-error). At a higher level, many lens mechanisms exist in the literature, as discussed in section 4.2. While we have argued that, to our knowledge, none of them simultaneously supports requirements R1,. . .,R4, comparing such mechanisms with our lens for specific use-cases and datasets is an important test for the *end-to-end* effectiveness of our proposal. We have not covered this point as obtaining (or replicating) implementations of such lenses is very challenging. This remains an important open point for future work – both for our proposal but also for all other volumetric lens proposals in the literature. In particular, none of the techniques in Table 4.1 were compared side-by-side against other techniques. We have performed three user evaluations involving specialists in airport baggage

security (subsection 4.5.1), pulmonology (subsection 4.5.3), and air traffic control (subsection 4.5.4). In all cases, users were not involved in this work, nor with other work of the authors. However, the set-up of these evaluations stays at the level of formative user experiments. To confirm and refine the obtained (positive) findings, more formal user studies are needed, which we plan to cover next.

4.7 Conclusions

In this chapter, we presented a new fish-eye-like context-and-focus lens that addresses the occlusion problems inherent in scalar volume rendering. The principle of our lens consists in first gathering (squeezing) rays so that they easily pass through occluding densities (given a user-specified opacity transfer function) and next scattering (fanning out) rays to best sample the target of interest. Our lens can be directly applied to any DVR raycaster and scalar volume dataset. Its main constraint is that the user should be able to find a viewpoint from which the target of interest, deep buried in the data, is at least slightly visible. We also present several modifications of the local rendering parameters within the lens (view direction, lighting parameters, opacity transfer function) that aim to both better separate the focus (lens) from the context (volume) and also allow more detailed examining of the target. Our lens is easy to use – all its parameters are controlled via direct mouse-and-keyboard interaction – and can be efficiently implemented atop of a standard GPU ray caster. Our lens is especially useful for highlighting structures of interest which are both deeply embedded in volumetric data and cannot be revealed by standard transfer function manipulations due to similar densities in the occluders and target. We demonstrate these points using five use-cases involving datasets from baggage detection, fluid visualization, air traffic control, and chest radiology, and DTI fiber tracts.

Several improvements to our proposal are possible, as follows. First and foremost, heuristics can be sought to link all our free parameters (lens size, focus depth, interpolation between focus and context) directly to the volume data, so the user interaction is minimized and therefore exploration efficiency is increased. Secondly, our lens could be extended to different types of volumetric datasets, such as multivariate (vector, tensor) fields. Last but not least, a formal wider-scale evaluation of how the lens addresses more specific tasks, and how it compares to existing tools for these tasks, such as other lens types, is a goal we aim to pursue next.

Chapter 5

Volume rendering on mobile devices (Virtual Reality, Augmented Reality, Mixed Reality)

This chapter presents the research I have started at the end of this thesis. We address the volume rendering challenges on mobile devices (Virtual reality, augmented reality, and mixed reality). Mobile devices are getting more and more popular across the population. Although their technical specifications can be totally different from one device to another, they are all becoming more powerful int terms of memory, CPU, GPU, and features.

5.1 Introduction

First of all, let us define and be more following terms: virtual reality, augmented reality, and mixed reality.

Virtual Reality (VR) immerses users in a fully artificial digital environment. In fact, This technology immerses users in a completely virtual environment that is generated by a computer. The most advanced VR experiences even provide freedom of movement – users can move in a digital environment and hear sounds. Moreover, special hand controllers can be used to enhance virtual reality experiences.
You need to wear a special VR headset to experience virtual reality. Most VR headsets are connected to a computer (Oculus Rift) or a gaming console (PlayStation VR) but there are standalone devices (Google Cardboard is among the most popular) as well. Most standalone VR headsets work in combination with smartphones: the user insert a smartphone into a headset, then wear this headset, and immerse in the virtual reality.

Augmented Reality (AR) allows users to see and interact with the real world while digital content is added to it. As a popular example, we can think of Pokemon Go which causes millions of people all over the world have been rushing with their smartphones in search for small virtual creatures. That is the most pertinent example of augmented reality.

If you own a modern smartphone, you can easily download an AR app and try this technology. There is a different way to experience augmented reality, though with special AR headsets, such as Google Glass, where digital content is displayed on a tiny screen in front of a user's eye.

Mixed Reality (MR) is the most recent development in reality technologies that sometimes causes confusion, primarily because different experiences are called so. Without going too deep into science, let us look at two forms of reality technologies that are referred to as mixed reality (As I have mentioned just one of them at the very beginning):

- **Mixed reality that starts with the real world** – virtual objects are not just overlaid on the real world but can interact with it. In this case, a user remains in the real-world environment while digital content is added to it; moreover, a user can interact with virtual objects. This form of mixed reality can be considered an advanced form of AR.
- **Mixed reality that starts with the virtual world** – the digital environment is anchored to and replaces the real world. In this case, a user is fully immersed in the virtual environment while the real world is blocked out. It almost looks like virtual reality. In fact it does, but the digital objects overlap the real ones whereas in conventional VR the virtual environment is not connected to the real world around a user. To experience this form of mixed reality, you can wear Windows mixed reality headsets.

5.2 Stereoscopic 3D

Stereoscopic 3D is used in virtual reality and mixed reality systems. This is a technique that produces an illusion of depth in a moving image by displaying two slightly different images to the right and left eye of the observer . This ability is based on the characteristics of the human visual system. The eyes, being positioned horizontally in the head, receive two views of the visual scene - one for the left-eye and another for the right-eye. The views overlap but differ slightly since they originate from two distinct perspectives. The visual system interprets and processes the information gathered from the two images to produce stereoscopic depth. The binocular system is very good at coordinating the movement of the eyes, which move constantly even during fixation. From a functional point of view, the images of both eyes fall on the fovea when fixating binocularly on a point.

The fovea is the part of the back of the eye that has the highest acuity. According to Tam et al., 2011, "an object fixated binocularly is imaged on the same relative coordinates in the left-eye and right-eye views and it is perceived as a single percept, i.e., it is seen as a single object."

To look at a new object located at a different distance, the point of fixation is altered. The two eyes move at the same time and in opposite directions so that the new object is imaged in the center of each eye's fovea. When the new object is closer

the eyes move inward (convergence). On the contrary, when the new object is farther away the eyes move outward (divergence). This process is called vergence and it is related to accommodation

5.3 Virtual Reality (VR)

There are two main types of VR headsets: PC-connected headsets and Standalone headsets. Each type influences differently the volume rendering process according to its specifications.

5.3.1 PC-connected headsets

As their name suggests, these VR headsets are connected to a computer (or a gaming console) that generates high-quality virtual experiences. The processing power of modern computers is huge, so they can generate realistic and persuasive digital worlds. This characteristics are really interesting for volume rendering since direct volume rendering is greedy in term of processing. The more processing capability available, the more quality we will get at the end of the rendering pipeline.

Another major constraint with virtual reality in general is the necessity to render two different image to create an illusion of depth thanks to stereoscopic 3D.

VR headsets can be used along with special controllers. In this case, users can actually interact with the virtual environment they are immersed in. As might be expected, PC-connected headsets provide the most engaging VR experiences.

The most popular PC-connected VR headsets are HTC Vive, PlayStation VR, and Oculus Rift.

5.3.1.1 Implementation

During this study, we implemented a volume renderer based on a cuda acclerated raycasting algorithm on a virtual reality headset. The device we used was the Oculus rift *Oculus Rift*. The performances on the occulus rift was half lower than the performances on a PC. This is due to the fact that all the computations are carried out by the CPU and the GPU of the PC to which the headset is connected. However, it is possible to increase the frame rates just by reducing the resolution or the quality of the rendered images. Figure 5.1 shows a lung CT scan rendered on the oculus rift using a CUDA accelerated raycasting algorithm. So far, we did not developed the interactions to manipulate and modify the volume inside the head mounted device. We still use the mouse even though it is not simple to use external devices while wearing an headset.

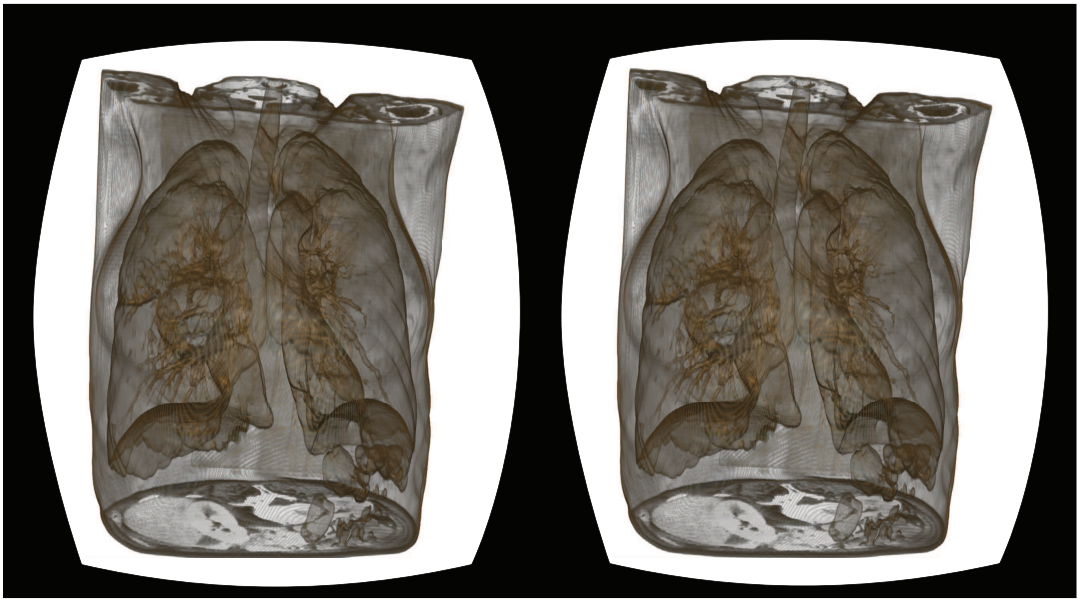


FIGURE 5.1: A lung CT scan rendered on the oculus rift using a CUDA accelerated raycasting algorithm

5.3.2 Standalone headsets

Until now, PC-connected VR headsets are quite expensive and relatively few people are willing to invest their money in them. Yet there is another way to experience virtual reality – using standalone headsets that do not need to be connected to a computer or console.

Most standalone VR headsets use a smartphone screen to provide the virtual reality experience. Such devices are quite affordable, as users can simply insert their smartphone into the headset to enjoy VR. Samsung Gear VR, Google Daydream, and Google Cardboard work exactly this way.

Other standalone headsets work on their own. Facebook's soon-to-be-released Oculus Go, for example, will need neither a computer nor a smartphone to generate virtual experiences. This device is likely to make virtual reality technology a lot more common and affordable than it is now.

This standalone headsets are then more susceptible to have a larger user population than the PC-connected ones. It is then really appropriate to investigate how the volume rendering process can be well implemented and adapted to this kind of devices. However, due to the less powerful graphical process unit available in theses standalone devices, we have to find a better trade-off between the frame rate and the quality of the rendered image.

5.4 Augmented Reality (AR)

Augmented reality (AR) is the overlay of digital content on the real-world environment. Virtual objects can be in various forms: images, videos, or interactive data.

In other words, if you see the real world supplemented with digital objects, that is AR. Imagine you want to buy a piece of furniture – a chair, for example. Augmented reality technology can help you check how different chairs will look in your room and pick the one that fits best.

So how can you bring AR experiences to life? There are two main ways: Portable devices and Smart glasses and AR headsets.

5.4.1 Portable devices

Augmented reality is the most accessible reality technology, as people can use their smartphones or tablets to run augmented reality applications. AR apps use a phone camera to capture the real world; virtual objects are then overlaid and users can see them on their smartphone screen.

That is how common AR apps work, the best example being Pokemon Go. Millions of people have used their smartphones to play this game and catch virtual Pokemons that they can only see on their smartphone screens.

Since smartphones are extremely affordable now, augmented reality could bring more experience into the visualization of volumetric datasets.

Before developing an augmented reality application on a smartphone, the first step is to be able to render the volume. In this early step of our study, we have just realised a prototype thanks to the Unity platform to display a 3D volume. This prototype only renderer allows rotation and zooming interactions. The raycasting algorithm was implemented using Cg/HLSL shader language. Our prototype does not perform any shading in the rendering process, see **??** .

5.4.2 Smart glasses and AR headsets

Another way to create AR experiences is to use special smart glasses or headsets. Unlike VR headsets, these AR glasses and headsets do not immerse users into a fully virtual environment but just add digital objects to the real world. With Google Glass, for example, digital data is projected right in front of the user's eyes.

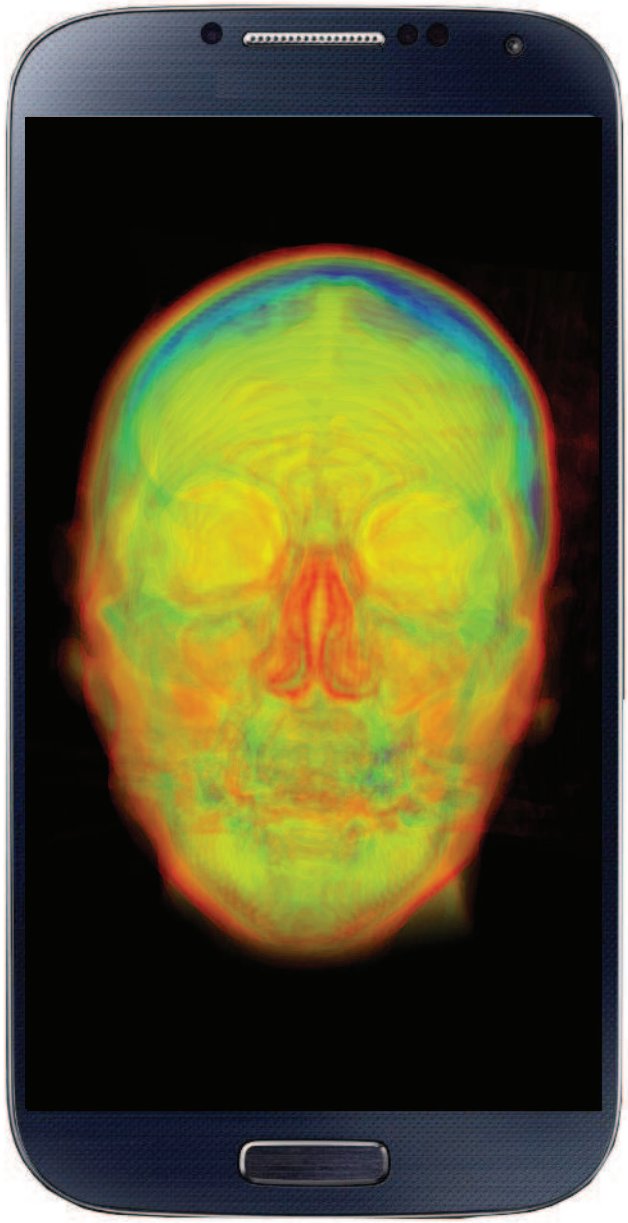


FIGURE 5.2: A head CT scan rendered without shading on an android smartphone

5.5 Mixed Reality (MR)

In mixed reality (sometimes called hybrid reality), virtual content is not only overlaid on the real environment (as in AR) but is anchored to and interacts with that environment.

In a nutshell, with mixed reality you can see virtual objects just like you can with augmented reality, but these objects can also interact with the real world. In a sense, mixed reality is a more immersive and interactive type of augmented reality.

There can be, however, a different form of mixed reality – when users see and interact with a completely virtual environment overlaid on the real world around them.

However, a common issue is to trip over a physical object in the room while interacting with a completely digital environment. To avoid this problem, a headset must be able to track the real world and adjust the virtual environment accordingly. This kind of mixed reality is closer to VR than AR; in fact, some VR headsets have sensors to track the physical environment too. Different types of devices are required to experience these two forms of mixed reality:

- **Holographic devices:** These headsets have translucent glasses that allow you to perfectly see your surroundings. Virtual experiences are created with the help of holograms. That is how Microsoft HoloLens works.
- **Immersive devices:** These headsets have non-translucent displays that completely block out the real world (just like VR headsets) and use cameras for tracking. Windows mixed reality headsets from Acer and HP work this way.

5.5.1 Implementation

During the end of this thesis, we begin to implement a volume rendering framework on a Holographic device. We use an Hololens (see *Microsoft Hololens*) as the holographic design to support the volume renderer framework. We tried two different types of implementations. The first one and the least difficult is to compute and render the volume on the holographic device itself. The second type of implementation is to use the **"remoting"** strategy. The remoting strategy allow to do all the computation on a computer (usually more powerful than the holographic device) and send the final images (one for each eye) through the WiFi to be rendered on the Hololens.

Another limitation of the *Microsoft Hololens* is the non compatibility with GPGPU langages such us CUDA and OpenCL which is quite obvious knowing the weak compute capability of the Hololens GPU called **HoloLens Graphics**. This is quite important for our framework since most of the novel and interesting interaction techniques rely on the computational power of the graphic card thanks to the GPGPU

approach.
To bypass this limitation, we used the **compute shaders** which are now available in the common shader langages (HLSL, GLSL).

The **compute shader** is a Shader Stage that is used entirely for computing arbitrary information. While it can do rendering, it is generally used for tasks not directly related to drawing triangles and pixels.

Compute shaders operate differently from other shader stages. All of the other shader stages have a well-defined set of input values, some built-in and some user-defined. The frequency at which a shader stage executes is specified by the nature of that stage; vertex shaders execute once per input vertex, for example (though some executions can be skipped via caching). Fragment shader execution is defined by the fragments generated from the rasterization process.

Compute shaders work very differently. The "space" that a compute shader operates on is largely abstract; it is up to each compute shader to decide what the space means. The number of compute shader executions is defined by the function used to execute the compute operation. Most important of all, compute shaders have no user-defined inputs and no outputs at all. The built-in inputs only define where in the "space" of execution a particular compute shader invocation is.

Therefore, if a compute shader wants to take some values as input, it is up to the shader itself to fetch that data, via texture access, arbitrary image load, shader storage blocks, or other forms of interface. Similarly, if a compute shader is to actually compute anything, it must explicitly write to an image or shader storage block.

5.5.1.1 Computation on the hololens

The GPU of the hololens is not really powerful and as a low memory capacity (around 600 MB). Knowing that the default and maximum supported resolution is 720p (1268x720) for each eye, have to compute two images with a lower resolution than this maximum value. To develop an holographic app, one can either use the Unity framework or directly develop an universal holographic application using Visual studio and directX.

Using the Unity framework to develop an holographic application is quite simple. We test different types of volume rendering algorithms: rayasting, 3D textures, and isosurfaces. We used Cg/HLSL in Unity to speed up the rendering process. Although the final result was beautiful, the prototype was not interactive because of low frame rates (between 1 and 5) according to the algorithm used and the different parameters such as the number of isosurfaces, the step during raycasting, or the number of slices computed when using 3D textures.

The second solution was to directly develop an universal holographic application using Visual studio and directX. It help us to gain more speed and frame rate than the unity version of each of these algorithms (see Figure 5.3, and Figure 5.4).

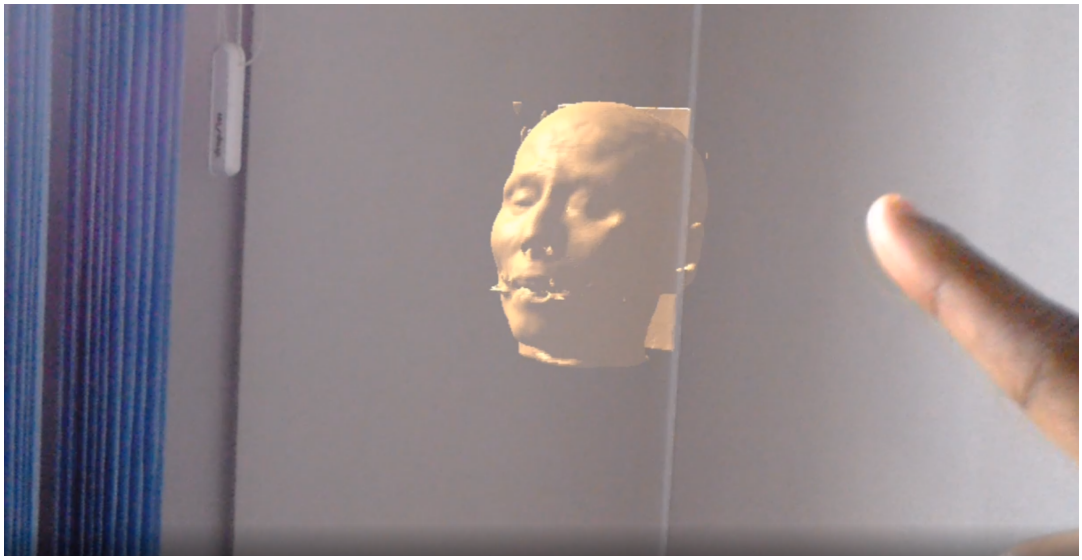


FIGURE 5.3: A head CT scan rendered on the hololens using isosurfaces

FIGURE 5.4: A head CT scan rendered on the hololens using a ray-casting algorithm

We just develop some basic interactions so far. For instance we switch from the isosurface representation to the one using the raycasting algorithm by using the **"air tap"** gesture. Air tap is a tapping gesture with the hand held upright, similar to a mouse click or select. This is used in most HoloLens experiences for the equivalent of a "click" on a UI element after targeting it. We use the **manipulation** gesture to modify the transfer function presets when using the raycasting algorithm.

5.5.1.2 Holographic remoting

Holographic remoting allows your app to target a HoloLens with holographic content hosted on a desktop PC or on a UWP device such as the Xbox One, allowing access to more system resources and making it possible to integrate remote immersive views into existing desktop PC software. A remoting host app receives an input data stream from a HoloLens, renders content in a virtual immersive view, and streams content frames back to HoloLens. The connection is made using standard Wi-Fi.

Holographic Remoting Player is a companion app that connects to PC apps and software that support Holographic Remoting. Holographic Remoting streams holographic content from a PC to your Microsoft HoloLens in real-time, using a Wi-Fi connection. The Holographic Remoting Player can only be used with PC apps that are specifically designed to support Holographic Remoting.

A typical remoting connection will have as low as 50 ms of latency. The player app can report the latency in real-time.

5.5.1.2.1 Performance

The Holographic remoting allows to have performances almost close to the ones when using a PC. I allows to bypass many weaknesses in the technical specifications of the Hololens. In fact, having access to the GPU and the CPU of a PC allow to perform a lot more heavy or time consuming tasks on this PC, before rendering the final result on the remote Hololens thanks to the Holographic Remoting Player. The quality and performance of your experience will vary based on three factors:

- The holographic application you are running - Applications that render high-resolution or highly-detailed content may require a faster PC or faster wireless connection.
- Your PC's hardware - Your PC needs to be able to run and encode your holographic experience at 60 frames per second. For a graphics card, Microsoft generally recommends a GeForce GTX 970 or AMD Radeon R9 290 or better. Again, your particular experience may require a higher or lower-end card.
- Your Wi-Fi connection - The holographic application is streamed over Wi-Fi. Therefore, it is important to a fast network with low congestion to maximize quality. Using a PC that is connected over an Ethernet cable, rather than Wi-Fi, may also improve quality.

Chapter 6

Conclusion

As seen in the previous chapters, the visualizations of volumetric datasets are not so trivial. These visualizations are used in various type of field such us medicine, physics, biology, archaeology, etc. These visualizations face different types of difficulties and challenges according to the application domain. For instance in art, the most important challenge is the beauty of the generated image while in most other application domains, the frame rate is also important.

In chapter 3, we studied the activity of the airport security agents in order to provide a relevant 3D exploration tool. Thanks to contextual interviews, we extracted the requirement for the new 3D system to replace efficiently the old 2 dimensional system. The existing 2D systems can suffer from 4 main dissimulation strategies. The first one is the **superposition** where a threat may be sheltered among dense materials. While possible to see through such a 'shield' using high penetration (enhanced X-ray power) or image processing (contrast improvement), such techniques are not universally available and also require fine-tuning many parameters, which slows down inspection. Second, the **location** can be used since objects located in the corners, edges, or in the luggage's frame are very hard to spot. Third, the **dissociation** allows to conceal a threat by spreading its parts in the luggage, *e.g*, by disassembling a weapon and scattering its parts. Finally, a **lure** can be used. In fact A minor threat (lure) like small scissors is clearly visible and catch the security agent's attention who can miss the real threat. In this chapter 3, we proposed an interactive visualization tool for 3D baggage inspection. This framework offers different types of interaction to perform a virtual inspection of baggage while dealing with occlusion issues.

In chapter 4 we focused on occlusion management strategies. In fact, occlusion is an issue in volumetric visualization as it prevents direct visualization of the region of interest. While many techniques such as transfer functions, volume segmentation or view distortion have been developed to address this, there is still room for improvement to better support the understanding of objects' vicinity. However, most existing Focus+Context fail to solve partial occlusion in datasets where the target and the occluder are very similar density-wise. For these reasons, we proposed a novel focus+context lens that fulfills simultaneously the four following requirements: Rapidly create an unobstructed view of the target (R1), allow a flexible local exploration of the target zone (R2), keep the context in which the target is visually

embedded (R3), and handle datasets where the target and occluders cannot be separated by transfer function manipulations (R4).

In chapter 5, we address the volume rendering challenges on mobile devices (Virtual reality, augmented reality, and mixed reality). Mobile devices are getting more and more popular across the population. Although their technical specifications can be totally different from one device to another, they are all becoming more powerful int terms of memory, CPU, GPU, and features. While **Virtual reality (VR)** immerses users in a fully artificial digital environment with devices such us HTC vive, **Augmented reality (AR)** overlays virtual objects on the real-world environment thanks to devices such us smartphones. In addition, **Mixed reality (MR)** not just overlays but anchors virtual objects to the real world so the user can interact with bot the real world and the virtual environment. We studied how to provide interactive volume rendering tools on each of type of these devices.

In this thesis we present two main contributions.

- First, we proposed a new interactive visualization system for 3D scanned baggage accelerated with GPGPU techniques in accordance with the needs we extracted from the contextual inquiry with the airport security agents.
- Secondly, we proposed a novel technique which combines high-quality DVR with a fast, versatile, and easy to use, lens to support the interactive exploration of occluded data in volumes.

6.1 summary

6.1.1 Design Study: Interactive exploration of 3D scanned baggage

During this thesis, I had the opportunity to intend to training courses for security agents' instructors, and to visit an airport (Toulouse-Blagnac Airport). These helped me to study the activity of security agents and get the users' needs. From these contextual interviews, we noticed the 4 main concealment strategies (superposition, location, dissociation, lure).

We propose a set of interaction accelerated by GPGPU computing especially with the Nvidia's CUDA API. The first interaction was the transfer function edition. Since airport security agents have a reduced time frame and limited knowledge of technical constraints, we defined six TF presets (Figure 3.10). These presets only modify the TF transparency curve while keeping the same color mapping. Secondly, we dealt with objects selection and investigation. In order to investigate in detail a specific object, we offered the possibility to interactively isolate it , remove surrounding items to address occlusion issues, or find a suitable point of view. For instance, the magic brushing removes all voxels with a lower density than the first one encountered at the beginning of the brushing process. This technique helps the user to directly define the densities he or she wants to brush. This technique avoid multiple

interactions with the histogram and its range slider to define the range of brushable densities.

Since we developed this system with four baggage security practitioners, we had the opportunity to validate the usefulness of each proposed technique. Through an interactive process, the users gave their feedback all along the development process which helped to assess and to guide the proposed interaction techniques. Feedback was mainly positive and the user did not face any specific difficulty to use our system. Users mainly appreciated the simple interface with few widgets and the reduced set of interactions. Surprisingly, the security agents instructors were very interested to use the transfer function. The histogram and its transfer function were also appreciated even if the corresponding technique is not simple to understand. We suppose that our interface motivates users to learn more regarding the technique behind it. The users also mentioned the need to display the actual density (the numerical value) of selected object. They also ask many times if the displayed color corresponds to the one currently in use in operational settings. This confirms the fact that users are willing to keep some existing features and prefer to use a system they are already familiar with. The users also appreciated our design requirements with smooth transitions and incremental investigations of baggage.

All of these observations and qualitative evaluations deserve to be validated through proper evaluations which are out of the scope of this design study paper. Our system is fully functional and interactive enough to perform baggage explorations. Many improvements can still be done in order to improve its usages in terms of new interactions and exploration time reductions. Technically speaking, the selection of the adequate point of view can be improved with more investigated faces. But in practice, this simple paradigm remains suitable. If the computed point of view is not fully satisfactory, one can manual rotate the baggage. Nevertheless, the developed algorithm will not necessary provide the best solution (the lowest occlusion) but a satisfactory one.

Our goal was to develop innovative interactions to support baggage exploration but we did not really took into account the optimization of the exploration duration time. Manipulating the 3D volume may take time as well as our new interaction techniques. We think that these techniques have a great potential but they are suitable to explore in more details a suspicious baggage. Existing investigation techniques (with the 2D flattened image) are suitable to quickly and efficiently detect "clean" baggage. Then our tools can be a good solution to further investigate a potential threat with more available time.

6.1.2 Design Study: Interactive exploration of 3D scanned baggage

In this chapter, we presented a new fish-eye-like context-and-focus lens that addresses the occlusion problems inherent in scalar volume rendering. The principle of our lens consists in first gathering (squeezing) rays so that they easily pass through occluding densities (given a user-specified opacity transfer function) and next scattering (fanning out) rays to best sample the target of interest. Our lens can be directly applied to any DVR raycaster and scalar volume dataset. Its main constraint is that

the user should be able to find a viewpoint from which the target of interest, deep buried in the data, is at least slightly visible. We also present several modifications of the local rendering parameters within the lens (view direction, lighting parameters, opacity transfer function) that aim to both better separate the focus (lens) from the context (volume) and also allow more detailed examining of the target. Our lens is easy to use – all its parameters are controlled via direct mouse-and-keyboard interaction – and can be efficiently implemented atop of a standard GPU ray caster. Our lens is especially useful for highlighting structures of interest which are both deeply embedded in volumetric data and cannot be revealed by standard transfer function manipulations due to similar densities in the occluders and target. We demonstrate these points using five use-cases involving datasets from baggage detection, fluid visualization, air traffic control, and chest radiology, and DTI fiber tracts.

Several improvements to our proposal are possible, as follows. First and foremost, heuristics can be sought to link all our free parameters (lens size, focus depth, interpolation between focus and context) directly to the volume data, so the user interaction is minimized and therefore exploration efficiency is increased. Secondly, our lens could be extended to different types of volumetric datasets, such as multivariate (vector, tensor) fields. Last but not least, a formal wider-scale evaluation of how the lens addresses more specific tasks, and how it compares to existing tools for these tasks, such as other lens types, is a goal we aim to pursue next.

Appendix A

Questionnaire

A.1 Information

Age: .....
Profession: .....
Years of professional experience: .....
Familiarity with 3D:

Novice	Adv. Beginner	Competent	Proficient	Expert
☐	☐	☐	☐	☐

A.2 Scenario 1: Random Observation

Evaluate the level of difficulty when using the lens randomly to detect an object by changing the depth of the lens:

Easy	Quite Easy	Neutral	Quite Hard	Hard
☐	☐	☐	☐	☐

Comments:.....

How do you estimate the value of this tool?

Bad	Not Bad	Neutral	OK	Good
☐	☐	☐	☐	☐

Comments:.....

Compared with the traditional tools of scanners, I find that the present tool has added value

Bad	Not Bad	Neutral	OK	Good

☐ ☐ ☐ ☐ ☐

Comments:.....

A.3 Scenario 2: Observation of an area of interest

Evaluate the level of difficulty of using the lens on a partially visible object

Easy	Quite Easy	Neutral	Quite Hard	Hard
☐	☐	☐	☐	☐

Comments:.....

How do you estimate the value of this tool?

Bad	Not Bad	Neutral	OK	Good
☐	☐	☐	☐	☐

Comments:.....

Compared with the traditional tools of scanners, I find that the present tool has added value

Bad	Not Bad	Neutral	OK	Good
☐	☐	☐	☐	☐

Comments:.....

A.4 Suggestions

Do you have any ideas for improving the tool?
Comments:
.....
.....
.....

Do you have scenarios in which this tool would be very useful?
Comments:

.....
.....
.....

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.

Amira, Abbes et al. (2008). "A segmentation concept for positron emission tomography imaging using multiresolution analysis". In: *Neurocomputing* 71.10. Neurocomputing for Vision Research Advances in Blind Signal Processing, pp. 1954–1965. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2007.10.026. URL: http://www.sciencedirect.com/science/article/pii/S0925231208001306.

Ard, T. et al. (2017). "NIVR: Neuro imaging in virtual reality". In: *2017 IEEE Virtual Reality (VR)*, pp. 465–466. DOI: 10.1109/VR.2017.7892381.

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

Brambilla, Andrea et al. (2012). "Illustrative flow visualization: State of the art, trends and challenges". In: *Visibility-oriented Visualization Design for Flow Illustration*.

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.

Burns, Michael and Adam Finkelstein (2008). "Adaptive Cutaways for Comprehensible Rendering of Polygonal Scenes". In: *ACM SIGGRAPH Asia 2008 Papers*. SIGGRAPH Asia '08. Singapore: ACM, 154:1–154:7. ISBN: 978-1-4503-1831-0. DOI: 10.1145/1457515.1409107. URL: http://doi.acm.org/10.1145/1457515.1409107.

Carpendale, M. S. T., D. J. Cowperthwaite, and F. D. Fracchia (1997). "Extending distortion viewing from 2D to 3D". In: *IEEE Computer Graphics and Applications* 17.4, pp. 42–51. ISSN: 0272-1716. DOI: 10.1109/38.595268.

Carr, Hamish, Jack Snoeyink, and Ulrike Axen (2000). "Computing Contour Trees in All Dimensions". In: *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '00. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, pp. 918–926. ISBN: 0-89871-453-2. URL: http://dl.acm.org/citation.cfm?id=338219.338659 (visited on 08/18/2015).

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, C. D. and K. L. Ma (2011). "Visibility Histograms and Visibility-Driven Transfer Functions". In: *IEEE Transactions on Visualization and Computer Graphics* 17.2, pp. 192–204. ISSN: 1077-2626. DOI: 10.1109/TVCG.2010.35.

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.

Deans, Stanley R (2007). *The Radon transform and some of its applications*. Courier Corporation.

Dragicevic, Pierre et al. (2011). "Temporal Distortion for Animated Transitions". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '11. Vancouver, BC, Canada: ACM, pp. 2009–2018. ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1979233. URL: http://doi.acm.org/10.1145/1978942.1979233.

Elmqvist, N. and P. Tsigas (2008). "A Taxonomy of 3D Occlusion Management for Visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 14.5, pp. 1095–1109. ISSN: 1077-2626. DOI: 10.1109/TVCG.2008.59.

Everts, M. H. et al. (2009). "Depth-Dependent Halos: Illustrative Rendering of Dense Line Data". In: *IEEE Transactions on Visualization and Computer Graphics* 15.6, pp. 1299–1306.

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.

Grady, L. and E. L. Schwartz (2006). "Isoperimetric graph partitioning for image segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.3, pp. 469–475. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2006.57.

Griebel, Michael et al. (2004). "Flow field clustering via algebraic multigrid". In: *Visualization, 2004. IEEE*. IEEE, pp. 35–42.

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.

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, B Tissoires, and S Conversy (2009). "FromDaDy: Spreading data across views to support iterative exploration of aircraft trajectories". In: *IEEE TVCG* 15.6, pp. 1017–1024.

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

Hurter, Christophe (2015). "Image-Based Visualization: Interactive Multidimensional Data Exploration". In: *Synthesis Lectures on Visualization* 3.2, pp. 1–127.

Hurter, Christophe, Ozan Ersoy, and Alexandru Telea (2012). "Graph bundling by kernel density estimation". In: *Computer Graphics Forum*. Vol. 31. 3pt1. Blackwell Publishing Ltd, pp. 865–874.

Hurter, Christophe et al. (2014b). "Interactive image-based information visualization for aircraft trajectory analysis". In: *Transportation Research Part C: Emerging Technologies* 47, pp. 207–227.

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.

Kruger, J., J. Schneider, and R. Westermann (2006). "ClearView: An Interactive Context Preserving Hotspot Visualization Technique". In: *IEEE Transactions on Visualization and Computer Graphics* 12.5, pp. 941–948. ISSN: 1077-2626. DOI: 10.1109/TVCG.2006.124.

Lhuillier, Antoine, Christophe Hurter, and Alexandru Telea (2017). "FFTEB: Edge Bundling of Huge Graphs by the Fast Fourier Transform". In: *PacificVis 2017, 10th IEEE Pacific Visualization Symposium*. IEEE.

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.

Li, Wei et al. (2012). "Luggage Visualization and Virtual Unpacking". In: *Proceedings of the Workshop at SIGGRAPH Asia*. WASA '12. Singapore, Singapore: ACM, pp. 161–168. ISBN: 978-1-4503-1835-8. DOI: 10.1145/2425296.2425325. URL: http://doi.acm.org/10.1145/2425296.2425325.

Lorensen, William E. and Harvey E. Cline (Aug. 1987). "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". In: *SIGGRAPH Computer Graphics* 21.4, pp. 163–169. DOI: 10.1145/37402.37422.

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.

Merzkirch, Wolgang (2012). *Flow visualization*. Elsevier.

Microsoft. *Microsoft Hololens*. https://www.microsoft.com/en-us/hololens.

Pascucci, Valerio, Kree Cole-McLaughlin, and Giorgio Scorzelli (2004). "Multi-resolution computation and presentation of contour trees". In: *Proc. IASTED Conference on Visualization, Imaging, and Image Processing*, pp. 452–290. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.308.8105&rep=rep1&type=pdf (visited on 08/18/2015).

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.

Preim, Bernhard and Charl P Botha (2013). *Visual Computing for Medicine: Theory, Algorithms, and Applications*. Newnes.

Preim, Bernhard et al. (2016). "A Survey of Perceptually Motivated 3D Visualization of Medical Image Data". In: *Computer Graphics Forum* 35.3, pp. 501–525. ISSN:

1467-8659. DOI: 10.1111/cgf.12927. URL: http://dx.doi.org/10.1111/cgf.12927.

Ran, L. and J. Dingliana (2016). "Infocarve: A Framework for Volume Visualization on Commodity Augmented Reality Displays". In: *2016 International Conference on Virtual Reality and Visualization (ICVRV)*, pp. 473–479. DOI: 10.1109/ICVRV.2016.86.

Rezk-Salama, Christof and Andreas Kolb (2006). "Opacity Peeling for Direct Volume Rendering". In: *Computer Graphics Forum* 25.3, pp. 597–606. ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2006.00979.x. URL: http://dx.doi.org/10.1111/j.1467-8659.2006.00979.x.

Rift, Oculus. *Oculus Rift*. https://www.oculus.com/.

Sahoo, P. K. et al. (Feb. 1988). "A Survey of Thresholding Techniques". In: *Comput. Vision Graph. Image Process.* 41.2, pp. 233–260. ISSN: 0734-189X. DOI: 10.1016/0734-189X(88)90022-9. URL: http://dx.doi.org/10.1016/0734-189X(88)90022-9.

Sonnet, Henry, Sheelagh Carpendale, and Thomas Strothotte (2004). "Integrating Expanding Annotations with a 3D Explosion Probe". In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. AVI '04. Gallipoli, Italy: ACM, pp. 63–70. ISBN: 1-58113-867-9. DOI: 10.1145/989863.989871. URL: http://doi.acm.org/10.1145/989863.989871.

Tam, W. J. et al. (2011). "Stereoscopic 3D-TV: Visual Comfort". In: *IEEE Transactions on Broadcasting* 57.2, pp. 335–346. ISSN: 0018-9316. DOI: 10.1109/TBC.2011.2125070.

Telea, A. and J. J. van Wijk (1999). "Simplified representation of vector fields". In: *Proc. IEEE Visualization*, pp. 263–270.

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. (2012). "Stacking-Based Visualization of Trajectory Attribute Data". In: *IEEE Transactions on Visualization and Computer Graphics* 18.12, pp. 2565–2574. ISSN: 1077-2626. DOI: 10.1109/TransactionsonVisualizationandComputerGraphics.2012.265.

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.

Traoré, M. and C. Hurter (2017). "Interactive Exploration of 3D Scanned Baggage". In: *IEEE Computer Graphics and Applications* 37.1, pp. 27–33. ISSN: 0272-1716. DOI: 10.1109/MCG.2017.10.

Tversky, Barbara, Julie Bauer Morrison, and Mireille Betrancourt (Oct. 2002). "Animation: Can It Facilitate?" In: *Int. J. Hum.-Comput. Stud.* 57.4, pp. 247–262. ISSN: 1071-5819. DOI: 10.1006/ijhc.2002.1017. URL: http://dx.doi.org/10.1006/ijhc.2002.1017 (visited on 10/02/2015).

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.

Wang Baldonado, Michelle Q., Allison Woodruff, and Allan Kuchinsky (2000). "Guidelines for Using Multiple Views in Information Visualization". In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. AVI '00. Palermo, Italy: ACM, pp. 110–119. ISBN: 1-58113-252-2. DOI: 10.1145/345513.345271. URL: http://doi.acm.org/10.1145/345513.345271.

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.

Wu, Y. and H. Qu (2007). "Interactive Transfer Function Design Based on Editing Direct Volume Rendered Images". In: *IEEE Transactions on Visualization and Computer Graphics* 13.5, pp. 1027–1040. ISSN: 1077-2626. DOI: 10.1109/TVCG.2007.1051.

Xin, Y. and H. C. Wong (2016). "Intuitive volume rendering on mobile devices". In: *2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 696–701. DOI: 10.1109/CISP-BMEI.2016.7852799.