EuroVis 2018
R. S. Laramee, G. E. Marai, and M. Sedlmair
(Guest Editors)

Volume 37 (2018), Number 3
STAR – State of The Art Report

Project Report: Direct Volume Rendering Techniques for Cardiovascular Visualization

Christian Brändle & Niko Leopold

Visualization of Medical Data 2, 2017W

Abstract

The visualization of cardiovascular structures is an essential aspect of the diagnostic process therapy planning. Here we want to apply a specific rendering technique called Maximum Intensity Difference Accumulation *(MIDA) [BG09] that is suitable to visualize structures like blood vessels without fine tuning complex transfer functions. MIDA is a combination of advantages from* Direct Volume Rendering *(DVR) and* Maximum Intensity Projection *(MIP).*
After listing our research in regards of applications and frameworks we describe the essential features of MIDA and conclude with some visualizations generated from our application.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—; 1.3.7 [Computer Graphics]: Three-dimensional Graphics and Realism—; 1.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—;

Keywords: Cardiovascular visualization, direct volume rendering, raytracing, transfer function design, intensity projection, vascular system modeling

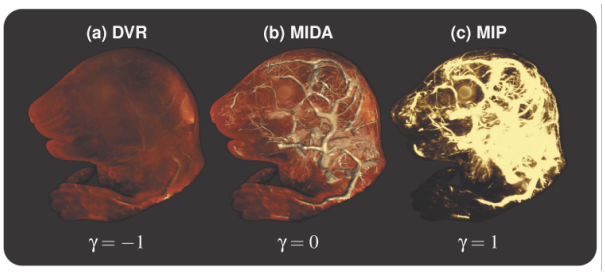

Figure 1: *Ultramicroscopy of a mouse embryo renderg using (a) DVR, (b) MIDA, and (c) MIP [BG09].*

1. Motivation

In the STAR report we have discussed various techniques related to the visualization of cardiovascular structures. We have covered both model-based and model-free approaches. Model-based mesh generation follows some more restrictive assumptions about the vascular structure whereas model-free methods such as direct volume rendering tries to capture as much original information as possible. Model-based techniques are often used for treatment planning and basic anatomical overview, whereas model-free methods are required for precise diagnostics.

We decided that for the project implementation we wanted to focus on direct volume rendering. The goal was to effectively visualize thin vessel structures and if possible even their narrowings due to stenoses etc. At first we considered to implement the automatic transfer function specification for visual emphasis of coronary artery plaque by Glasser et al. [GOH*10]. However we realized that due to time constraints this would not be possible as it involves a large number of different pipelines stages. We thus decided to implement instant volume visualization using *Maximum Intensity Difference Accumulation* (MIDA) by Bruckner and Gröller [BG09].

2. Data

To visualize cardiovascular data we first had to search approprite data that exhibit the information we want to visualize. We refer here to DICOM and DAT files, as the former are relevant in medical visualization and the latter are made out of our own format that is read by our application.

2.1. DICOM

In the field of medical visualization DICOM is the defacto standard for capturing, annotating and deploying data. Including meta information and high dynamic range images DICOM is our preferred way to load medical data into our system.

2.2. DAT

The DAT format consists of 16 bit data chunks where the first three chunks provide the resolution of the volume in x,y and z-coordinate

for uniform scaled voxels. The following data entries also consist of 16 bit chunks where the values are are encoded as 12 bit values.

An extended format of DAT has 16 bit data chunks for x,y,z and for the number of bits in the following data section. In that way space is more efficiently used, but cause us to provide another conversion step.

2.3. Data Acquisition

To our disappointment we had to find out that it is rather difficult to find appropriate vascular data of the heart. Important data sources like the *Osirix DICOM image library* [osi18] are of commercial nature and as such of no use for us.

2.4. Data Import

One of the key advantages of DICOM data over simple image stacks is the fact that pixel intensities and radiometric intensities do not need to be the same in a given image stack. So DICOM provides a mapping to use several images together within a homogen radiometric background.

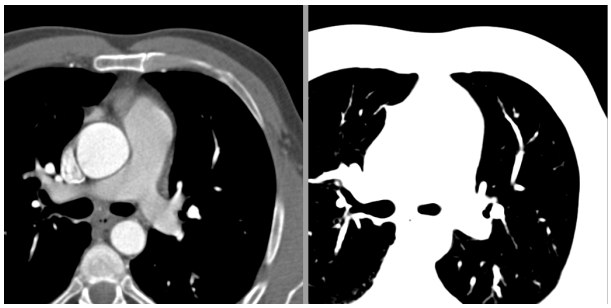

Figure 2: *Stacked images with inhomogenous radiometric space. DIOCM sample data* CARDIX *[gim18a] aquired from sample data section of the GIMAIS project [gim18b] and processed with* dicom2jpeg *from imbera project [ime18].*

Simple exports of JPEG data out of DICOM without respecting this relation spills out images with are not homogenous across the stack as seen for instance in figure 2.

2.5. Data Conversion

As it was not possible to find a suitable DICOM reader for our *CARDIX* dataset [gim18a] for VTK we used an intermediate step via MITK and convert DICOM data with the help of MITK main application to VTK 3D Image (VTI) data readable by VTK. In this way we could overcome the problems with codecs and intensity differences across the image stack.

Later we convert the data from VTI file format to our own DAT format that holds 12 bit intensity values in row major order where every data entry consists of an unsigned short, so 16 bit value. The first three entries describe the resolution of the dataset in x,y and z direction. Alternatively we also convert to the extended DAT format as described in section 2.2.

For the *CARDIX* dataset we could convert the VTK image just with a regular rescaling of every data entry to our 12 bit range. For the *MAGIX* dataset we had to resample the original dataset. As scanning resolution can differ according to the dimension, we rescaled our 3D datasets to uniform resolution accordingly.

Alternatively we also convert from PVM file format defined by Roettger et al. [Roe18a] to DAT format to visualize the *Angio* dataset form the repository of *The Volume Library* [Roe18b] website.

2.6. Datasets

Here we list the datasets we visualize to show the abilities of MIDA.

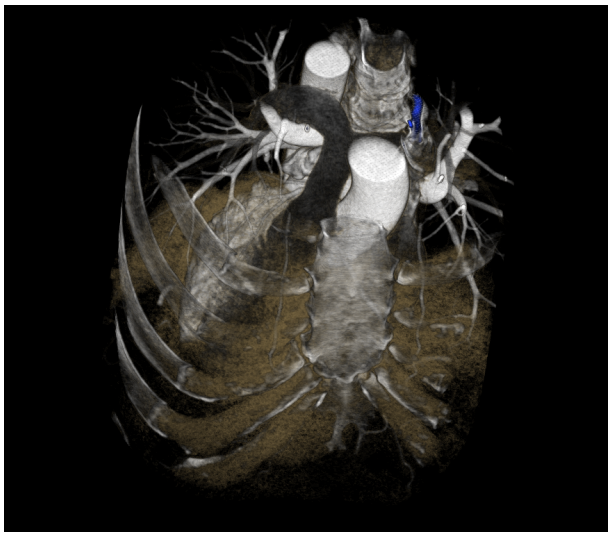

Figure 3: *DAT conversion from sample data* CARDIX *[gim18a] acquired from sample data section of the GIMAIS project [gim18b].*

Our datasets consist of *CARDIX* as shown in figure 3, *MAGIX* as shown in figure 4 and *Angio* as shown in figure 5.

3. Medical Visualization System

As our original goal was to use an existing application and enhance it with certain abilities we searched the web to find open source applications that are cross platform and run under Windows and Linux environments. As this is a quite restrictive approach we actually found a lot more applications and frameworks but exclude them for the reasons above.

Our search continued then with frameworks as we saw that there was no application that fulfilled our needs and was capable for rapid prototyping our system. With frameworks we unfortunately also didn't have more luck as the way the frameworks limit us in regards of data import and data visualization was as well not suitable to develop our chosen algorithm rapidly.

In the end we utilize an OpenGL-based visualization project to implement our algorithm there. Additional libraries to load DICOM data were searched but also there we experience several problems with codecs for instance.

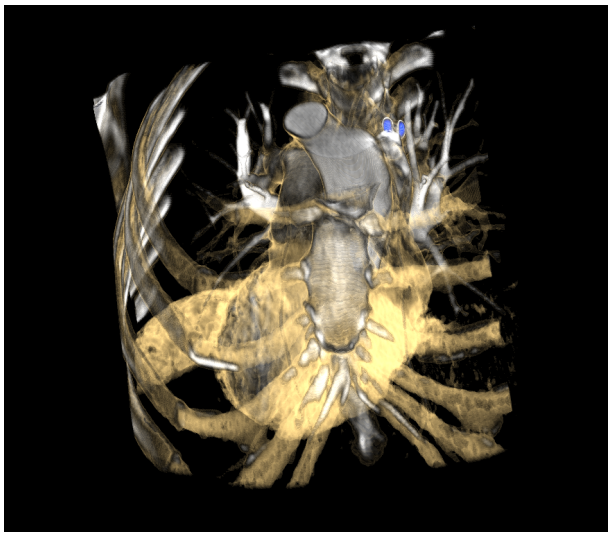

Figure 4: *DAT conversion from sample data* MAGIX *[gim18a] acquired from sample data section of the GIMAIS project [gim18b].*

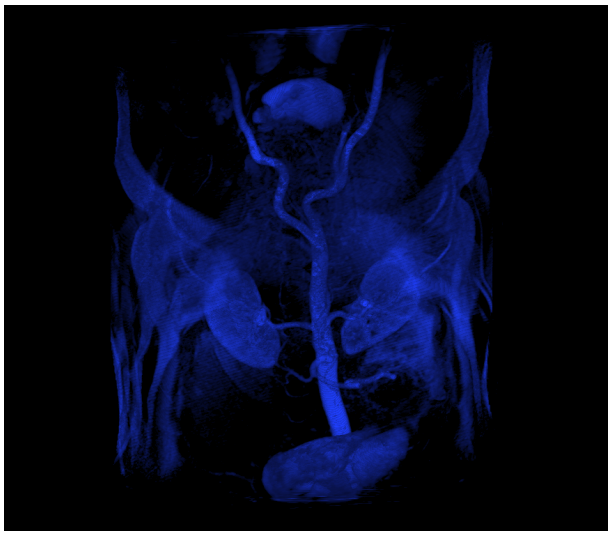

Figure 5: *DAT conversion from sample data* Angio *[Roe18b] acquired from* The Volume Library *website [Roe18b].*

3.1. 3DSlicer

„3D Slicer is an open source software platform for medical image informatics, image processing, and three-dimensional visualization [3DS18].“

The power of 3DSlicer lies in the handy user interface and the amount of resources how to deal with the program. Also youtube is a good source and where we lernt to do some basic tasks with 3DSlicer. The tool can also import several kinds of DICOM data, which is a prerequisite for our medical visualization application.

Although very nice at user level we do not discover a developer guide that guides us well through the internals of 3DSlicer. As our goal was not to use the application but rather to implement renderers we hadn't the impression that this would be possible in reasonable amount of time.

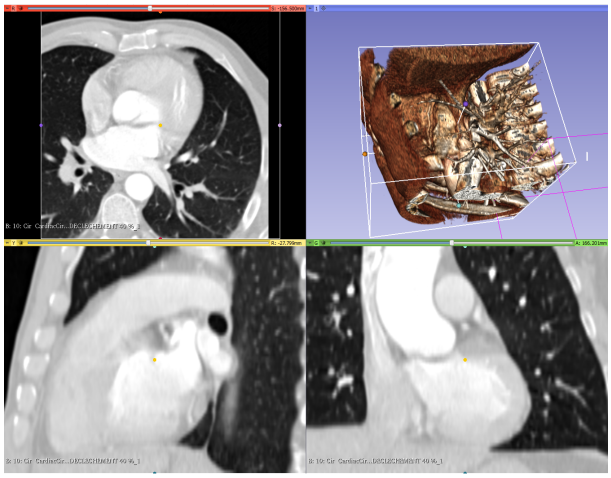

Figure 6: MAGIX *[gim18a] dataset visualized with* Slicer3D.

As could be seen in figure 6, we visualized one of our example datasets, *MAGIX* with volume rendering provided by 3DSlicer.

3.2. MITK - The Medical Imaging Interaction Toolkit

> The Medical Imaging Interaction Toolkit (MITK) is a free open-source software system for development of interactive medical image processing software. MITK combines the Insight Toolkit (ITK) and the Visualization Toolkit (VTK) with an application framework [MIT18].

In comparison to 3DSlicer MITK offers a better developer documentation and a description of the internal structure of MITK itself. Further it supplies guides how to create plugins or template-based projects that help to develop own applications with the MITK framework. Build instructions and a description of the render concept concludes our decision to take a closer look at this open source project. Also DICOM loaders are integrated and we managed to visualize one of our demo datasets in MITK. A downside was the stability of MITK as it happened several times that the application crashed after selecting DICOM data for volumetric visualization.

As we are interested in writing an own renderer and as MITK uses VTK to visualize data, we refer here to the VTK section 3.3 that describes our experience with VTK in more detail.

As could be seen in figure 7, we visualized one of our example datasets, *CARDIX* with volume rendering provided by the MITK application.

3.3. VTK - The Visualization Toolkit

> The Visualization Toolkit (VTK) is an open-source, freely available software system for 3D computer graphics, image processing, and visualization. It consists of a C++

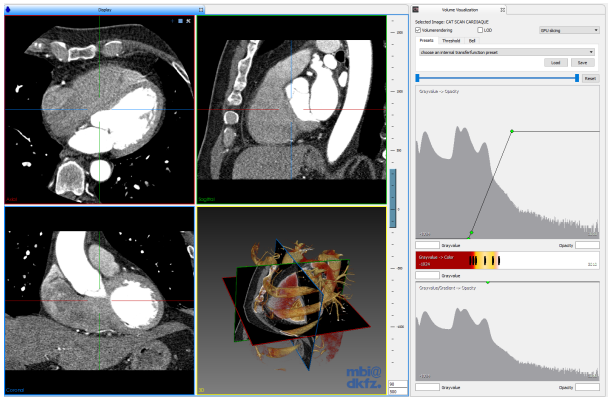

Figure 7: CARDIX *[gim18a] dataset visualized with* MITK.

class library and several interpreted interface layers including Tcl/Tk, Java, and Python. VTK supports a wide variety of visualization algorithms including scalar, vector, tensor, texture, and volumetric methods, as well as advanced modeling techniques such as implicit modeling, polygon reduction, mesh smoothing, cutting, contouring, and Delaunay triangulation. VTK has an extensive information visualization framework and a suite of 3D interaction widgets [vtk18].

As VTK was the central part of actual all medical visualization toolkits researched we wanted to use this framework for our application. Our resources for learning VTK were *The Visualization Toolkit* [SLM04], *The VTK User's Guide* [ABB*10] and of course the VTK webpage [vtk18].

Several parts of the VTK chain were researched, as there are:

3.3.1. Loading Data with VTK Readers

VTK also supports to load DICOM data via specific VTK readers. As our data is formed of DICOM files we take a closer look at that feature. Unfortunately, due to problems with the codec included in our DICOM data, we were not able to load datasets as we want to. As we haven't found a solution to fix this we come up with a preconversion step of DICOM data with MITK, as there it is possible to save DICOM data into a VTK 3D image.

3.3.2. Processing Data with VTK Filters

The VTK way of processing data is through VTK filters. The philosophy states that filters transform or combine data within a chain or network of coupled filters that finally provide an output to one or more VTK mappers. To implement our direct volume rendering approach the only easy way would have been to dynamically adapt our DICOM data according to the settings of MIDA and the current view direction and further use a built-in volume renderer like RayCast mapper to visualize the result. As we would have to implement somehow a reverse version of MIDA and as the data transformation would have been a separate step in the processing chain we decided not to use VTK filters at all.

3.3.3. Render Data with VTK Mappers

In VTK renderers are called mappers as they provide the ability to map a given dataset to a given device. While it is possible to override specific mappers it is not straight forward to interfere with the renderer at OpenGL-level.

For this reason VTK introduced VTK shaders, that are based on structured OpenGL shaders. While this gives access to GLSL shaders the structure of the shaders has to follow the VTK guidelines. That include string replacement of VTK specific shading commands within the OpenGL shader.

As we haven't found appropriate information on what for string replacements exist, how they should be used and how they interact with the rest of the VTK application, we decided not to use an OpenGL shader based on the VTK framework.

3.3.4. VTK Example Application

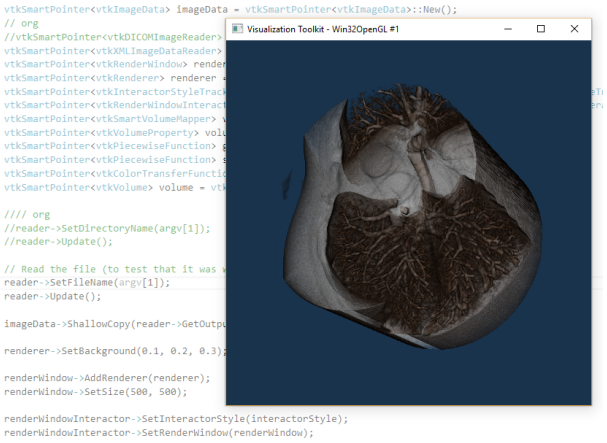

Figure 8: CARDIX *[gim18a] dataset rendered from VTI file with VTK RayCast Mapper.*

As VTK provides a lot of examples and tutorials how to use its framework we gave it a try and load a 3D image in form of a VTI file into a simple ray caster visualization. In figure 8 we see the *CARDIX* [gim18a] dataset visualized with this base facilities. As VTK also provide a script interface we also build Tcl/Tk, found at the *Tcl developer site* [tcl18], and tried the same visualization on a script base, which works perfectly fine. But as already stated before we left VTK behind as it does not support an easy integration of own OpenGL shaders.

3.4. Imbera DICOM Reader

„Imebra SDK is an open source C++ library that handles DICOM network messages and DICOM files [ime18].“

While Imbera seems to be the library we searched for to easily read DICOM data on multiple platforms in practice it fails because of the codec used in our *CARDIX* dataset. We don't know if this is due to the open source version used as we haven't tried the commercial version and will leave this as an open point.

3.5. OpenGL Application

As stated before we considered using the VTK (Visualization ToolKit) framework as well as the MITK (Medical Imaging Interaction Toolkit) framework. However we soon realized that it is hard to implement a custom volume renderer in these frameworks, and thus decided to fall back to our own implementation. For this we reused an old Qt framework from the *Visualization 1* course that had simple GPU direct volume rendering implemented with simple alpha compositing, but with no proper interaction, no transfer functions and no shading.

The goal was to extend this framework with

- More compositing methods: Maximum Intensity Projection (MIP), Minimum Intensity Projection (MINIP), Average Intensity Projection (AIP) as well as Maximum Intensity Difference Accumulation (MIDA)
- Interactive controls
- Gradient-based shading
- Customizable transfer functions

4. MIDA - Maximum Intensity Difference Accumulation

MIDA is a compositing technique for Direct Volume Rendering (DVR) and was first published from Bruckner et al. [BG09]. It combines the advantage of Maximum Intensity Projection (MIP), where important structure shine through all data, with the advantage from traditional DVR, where depth cues stem from color and opacity accumulation.

The advantages and disadvantages of the combined technologies are more precisely:

- Maximum Intensity Projection (MIP)
  + MIP does not require a transfer function .
  - The spatial context in MIP visualization is lost as the most important pixel just shine trough.

- Direct Volume Rendering (DVR)
  + DVR accumulates color and opacity, it uses mostly gradient-based shading.
  - Shading have to interpolate between unshaded and gradient-based shading according to gradient magnitude.
  - DVR needs appropriate transfer function, otherwise crucial information won't be visualized well and important features will disappear in fog or will be occluded by unimportant data.

4.1. MIDA - Accumulation

MIDA is rendered with front-to-back traversal order. It focus the interest on the regions of a ray where maxima are and values change from low to high. The amount of change is represented by δ_i for the i-th sample position P_i along the ray and f_{max_i} represents the current maximum found. MIDA overrides the occlusion relationship of the previous accumulated color c_i and opacity α_i when a new maximum is encountered, where the weights are defined by β_i.

$$\delta_i = \begin{cases} f_{P_i} - f_{max_i}, & \text{if } f_{P_i} > f_{max_i} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

$$\beta_i = 1 - \delta_i \tag{2}$$

$$c_i = \beta_i c_{i-1} + (1 - \beta_i \alpha_{i-1}) \alpha(f_{P_i}) c(f_{P_i}) \tag{3}$$

$$\alpha_i = \beta_i \alpha_{i-1} + (1 - \beta_i \alpha_{i-1}) \alpha(f_{P_i}) \tag{4}$$

It has to be noted that color c_i and opacity α_i are computed the same way as for normal DVR, except that there is the additional weighting with β_i, see equation (4).

4.2. MIDA - Interpolation

MIDA can interpolate between it's two extrema, namely DVR and MIP, in a continuous fashion. To achieve this a interpolation variable γ is defined that ranges from -1 to 1, i.e. $\gamma = [-1, 1]$. As interpolation is different in the direction of DVR compared to the direction of MIP we treat each case accordingly.

4.2.1. MIDA to DVR

As explained in section 4.1 interpolation is done in 3D domain. It is based on the modulation of previously accumulated color and opacity. As γ moves from MIDA to DVR we adjust the value of β accordingly.

$$\beta_i = \begin{cases} 1 - \delta_i(1 + \gamma), & \text{if } \gamma < 0 \\ 1 - \delta_i, & \text{otherwise} \end{cases} \tag{5}$$

In that way equation (5) smoothly fade out high intensity values while γ is reduced.

4.2.2. MIDA to MIP

Because regions of interest would be 'thinned' to just only one value, shading in 3D domain would lead to artifacts when γ is arriving at a value of 1. So in contrast to section 4.2.1 interpolation here is done in image domain between color and opacity values of MIDA and color and opacity values of MIP.

4.3. MIDA - Shading & Classification

As shading exposes artifacts if it is based on gradient directions with low magnitude, like for instance gradients from noise, the magnitude of gradient is used to interpolate between shaded and unshaded color.

To classify, separate, highlight or point out certain parts of the dataset MIDA uses simple transfer functions. The used kind of transfer function is limited to brightness and contrast adjustment using the common window/level approach. Alternatively users can also select values from color maps or the like if they have preferences on that.

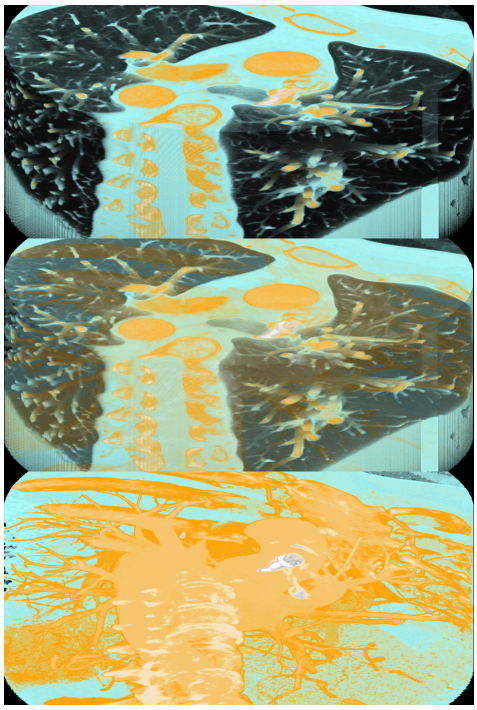

Figure 9: CARDIX *[gim18a] dataset visualized with* MIDA. *From top to bottom: DVR, MIDA, MIP.*

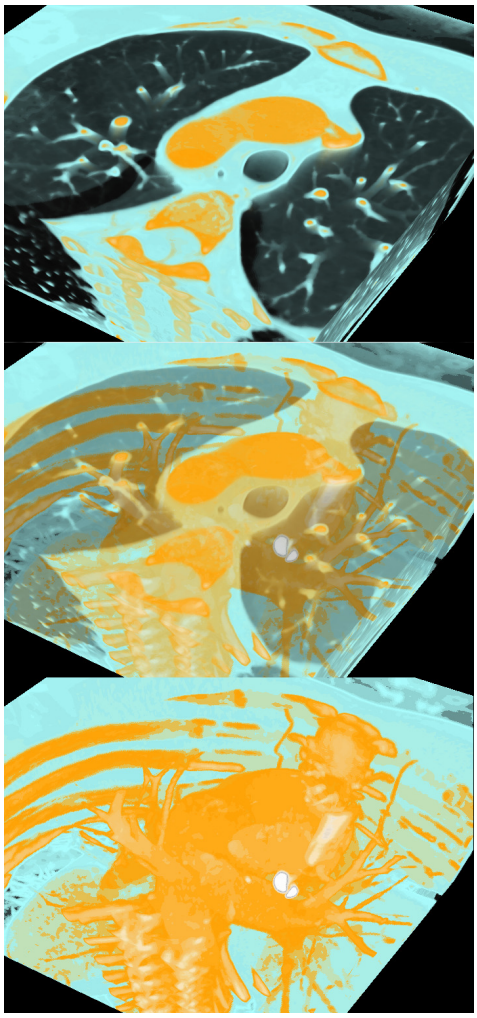

Figure 10: MAGIX *[gim18a] dataset visualized with* MIDA. *From top to bottom: DVR, MIDA, MIP.*

5. Example Renderings

Here we want to list some example renderings that show the capability of MIDA.

In figure 9 and 10 we can see the advantage of MIDA compared to DVR and MIP rendering. MIDA highlights important structures while not loosing the impression of depth in corresponding images.

Figure 11 and figure 12 shows several transfer function settings at the same MIDA-level. As seen certain coloring greatly enhance the visual perception of blood vessels and separate them better from background information.

Finally figure 13 shows another collection of renderings. From left to right and top to bottom we have a DVR and MIP rendering in the top row followed by two MIDA renderings with different settings in the bottom row. Again the better visualization of vessel structures is quite good visible.

6. Conclusion

The lessons learned from our *Medical Visualization 2* project were mainly about careful considerations necessary when using a framework. While for a regular project the usage of a framework will pay off soon, it is different for a project of this size.

First we were enthusiastic using applications made for medical visualization like *3DSlicer* and *The Medical Imaging Interaction Toolkit* (MITK). Their features are tremendous and they also provide possibilities to infer with the program at script or code level. There also exists nice dialogs for interaction, like transfer function dialogs and the like. But the structure of the applications also limit the flexibility of what can be implemented and how it has to be implemented. We were not able to implement our OpenGL rendering algorithm in such a framework without spending too much time on understanding the internals for the system. While this doesn't mean that it is impossible to extend the rendering capabilities of those applications we are convinced that we would have spent too much time on understanding the internals and therefore that this track was too dangerous with our deadline in mind.

Then we inspect the *Visualization ToolKit* (VTK), actually for two reasons. First VTK is the integral part of more or less all medical visualization applications observed and second VTK itself is also a good starting point for crafting own visualization applications. But soon we realized that also in VTK it is difficult to implement an own OpenGL-based renderer. More suitable for modifying data in a chain or network of filters, VTK also put heavy restrictions on

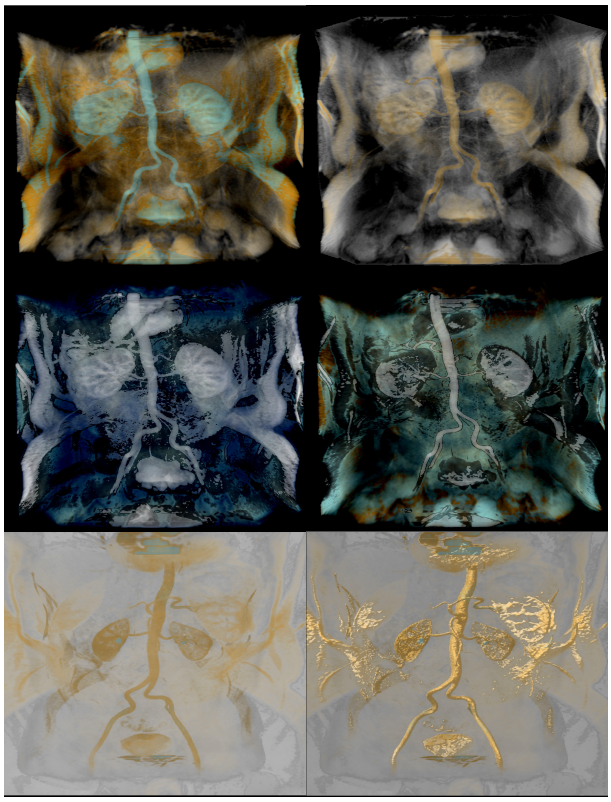

Figure 11: Angio *[gim18a] dataset visualized with* MIDA. *For each picture different transfer function settings are used.*

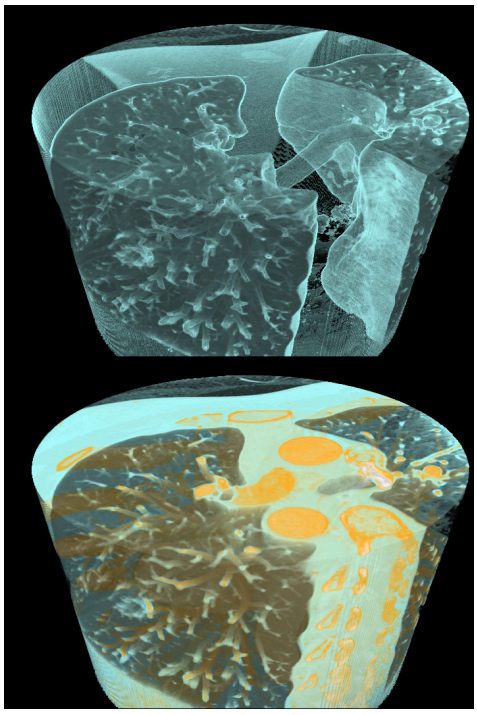

Figure 12: CARDIX *[gim18a] dataset visualized with* MIDA *with different thresholds applied in transfer functions.*

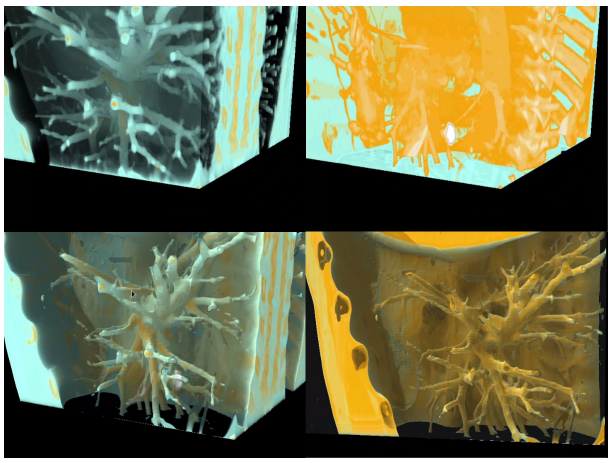

Figure 13: MAGIX *[gim18a] dataset visualized with* MIDA. *Top row (from left to right) shows DVR and MIP rendering. Bottom row shows MIDA with different thresholds applied in transfer functions.*

the creation of OpenGl-based renderers. The elements that should be used in a VTK OpenGL renderer are strings that are replaced internally to assemble a final version of an OpenGL shader. While we understand the flexibility of that approach it seems to be risky to use a weakly documented replacement strategy inside an OpenGL shader to accomplish our task at hand.

In regards of our DICOM files we experienced that they were not easily loadable by VTK or *Imbera* because of codec issues. We assume that freely available DIOCM data intentionally suffers from strange codecs and low resolution to motivate people to buy commercial DICOM data instead. Especially for cardiovascular data we actually only found one source at *GIMIAS* [gim18a] sample data section that supply such data for free. To read our data we did a workaround via MITK and VTK, namely we export our DICOM data in MITK as a 3D VTK image and import that VTI file into our own application. That way we could solve the codec problem. Other PVM-based datasets that stem from *The Volume Library* [Roe18b] need similar treatment.

In the end we relied on our own implementation that was easier to extend for the task at hand. The only challenge was to include our DICOM data because of the reasons explained before. Other than that the task of implementing MIDA was quite straight forward.

References

[3DS18] 3d slicer. https://www.slicer.org/, Jan 2018. 3

[ABB*10] AVILA L. S., BARRE S., BLUE R., GEVECI B., HENDERSON A., HOFFMAN W. A., KING B., LAW C. C., MARTIN K. M., SCHROEDER W. J.: *The VTK User's Guide*. Kitware New York, 2010. 4

[BG09] BRUCKNER S., GRÖLLER M. E.: Instant volume visualization

using maximum intensity difference accumulation. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 775–782. 1, 5

[gim18a] Gimias sample data. https://sourceforge.net/projects/gimias/files/SampleData/, Jan 2018. 2, 3, 4, 6, 7

[gim18b] Graphical interface for medical image analysis and simulation. http://www.gimias.org/, Jan 2018. 2, 3

[GOH*10] GLASSER S., OELTZE S., HENNEMUTH A., KUBISCH C., MAHNKEN A., WILHELMSEN S., PREIM B.: Automatic transfer function specification for visual emphasis of coronary artery plaque. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 191–201. 1

[ime18] Imbera - open source c dicom library /windows linux ios os-x android. https://imebra.com/, Jan 2018. 2, 4

[MIT18] The medical imaging interaction toolkit (mitk). http://mitk.org/wiki/MITK, Jan 2018. 3

[osi18] Dicom image library. http://www.osirix-viewer.com/resources/dicom-image-library/, Jan 2018. 2

[Roe18a] ROETTGER S.:. http://paulbourke.net/dataformats/pvm/, Jan 2018. 2

[Roe18b] ROETTGER S.:. http://schorsch.efi.fh-nuernberg.de/data/volume/, Jan 2018. 2, 3, 7

[SLM04] SCHROEDER W. J., LORENSEN B., MARTIN K.: *The visualization toolkit: an object-oriented approach to 3D graphics*. Kitware, 2004. 4

[tcl18] Tcl developer site, Jan 2018. 4

[vtk18] The visualization toolkit (vtk). https://www.vtk.org/, Jan 2018. 4