Data Visualization – Course Work

Assignment 5

Quick Facts

- Deadline: **Tuesday, Jan 29, 11:59:59! Upload only source code and pics**
- 3 tasks: Volume rendering in ParaView, C++ app with VTK, lecture recap on marching squares
- Submission via OPAL as one single ZIP file. Write names of team members in file name
- You should solve the tasks in groups of two
- Be prepared to present your answers orally during the course
- There is a tutor to answer your questions every Friday, 2nd DS, E042

Introduction

Please copy the source files into the existing folder structure so that "assignment5" with all of its subfolders are located next to "assignment4". You should re-use the folders "data" and "vtk" from the previous assignment. We provide a new data set called "headsq-half.vti", a CT image of a person's head and "ctTorso.vti", a CT scan of an upper female body.

Important note: Please upload **only** the source code files. No build files, no libs, no additional files. Please do not use any system-specific routines. Everything should compile under Visual Studio out of the box.

Task 1: Volume Visualization with Paraview

Description: Get in touch with ray-casting and isosurfaces.

- Import the dataset `/data/headsq-half.vti`
- Change representation from "outline" to "volume" and hit "apply" (use the volume rendering mode smart)
- Use the color transfer function editor to play around an get a feeling for this kind of visualization
- Create 2 screenshots where you can see the rendering (e.g. a special part of the CT data) as well as the transfer function (cp. Fig 1-1)
- Which part of the body has the highest local change in density? Which filter can you utilize to solve this question?
- What is the density value for bones? Use the contour filter and try to find a good value in order to produce a clean mesh. Make a screenshot (cp. Fig. 1-2)
- **Deliverable:** PDF file with your answers/explanations and screenshots.

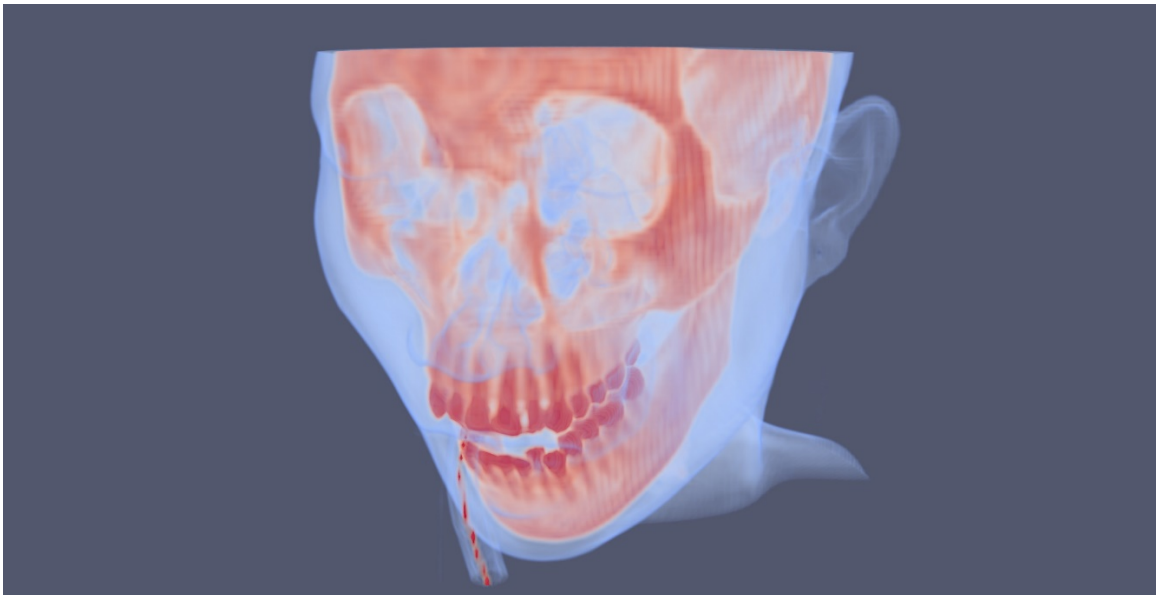

Figure 1-1. Direct volume rendering of CT scan in 3D.

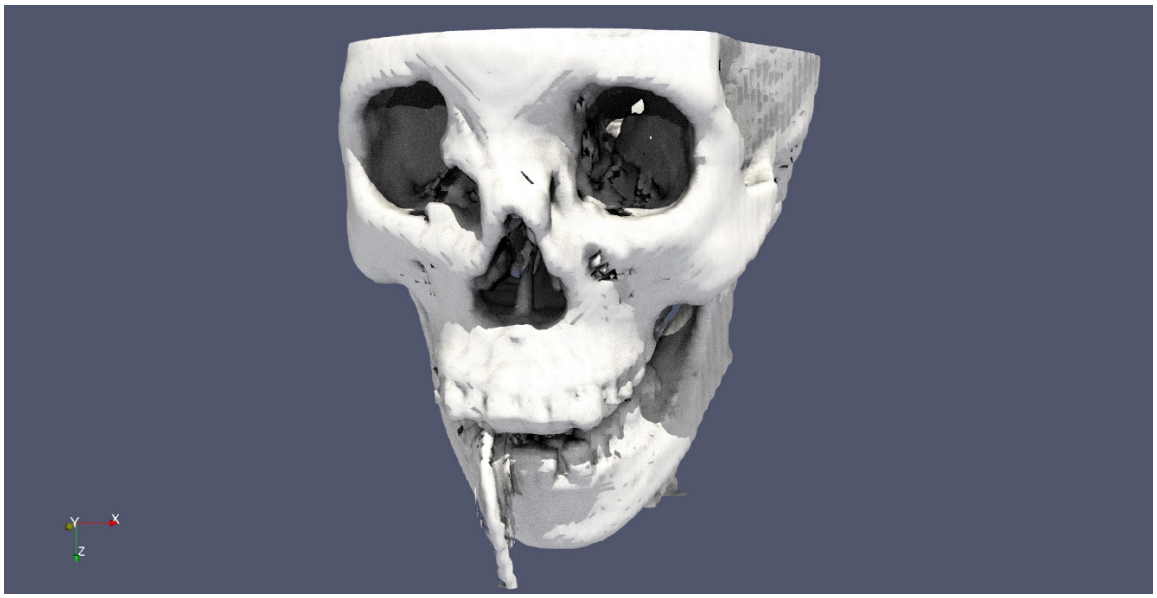

Figure 1-2. Isosurface of CT scan in 3D.

Task 2: Volume Visualization with VTK

Description: Assignment5.cpp has lots of useful hints in the code on how to use the classes and how to put everything together. Furthermore there are some helper methods implemented in vtkhelper.h/cpp that you can use.

Now the task is to combine two different approaches to volume rendering,

a) direct volume visualization and

b) contour rendering

into one single output image (see Fig. 2-1).

Provide a slider in order to change the iso-level interactively. For that you will need your own callback as a derived class of `vtkCommand` which is already given in the code.

- Given the data reader use a `vtkSmartVolumeMapper` in order to render the volume data directly
  - Turn on GPU-based rendering for high performance
  - set blending mode to **composite**
  - design an opacity transfer function via `vtPiecewiseFunction` with at least 2 density-opacity pairs
  - design a color transfer function via `vtkColorTransferFunction` with at least 3 density-color pairs
  - this time use `vtkVolume` instead of `vtkActor`
  - utilize `vtkVolumeProperty` for even more properties, apply it to the volume via `SetProperty`
  - create a renderer and a render window, test your code with `doRenderingAndInteraction(window)` method from `vtkhelper.h`
  - if everything went fine you can delete the line `doRenderingAndInteraction(window);` afterwards

- Put marching cubes on top
  - use `vtkMarchingCubes` filter
  - configure the filter that one meaningful iso-level will be rendered (e.g. use the value from task 1)
  - build the remaining pipeline: create mapper, actor and renderer and connect everything (use your knowledge from assignment 4)
  - test your intermediate result with `doRenderingAndInteraction(window)` like you did before

- create an interactive iso-value slider
  - create a slider 2D representation, add a title and set min/max values
  - create an interactor and assign the render window
  - assign representation and interactor to a new slider widget

  - use the given callback class to make the slider observable, try to understand the callback code and the observer pattern
  - start your combined model with `doRenderingAndInteraction(interactor)` from `vtkhelper.h`

- **optional**: there is a second data set "ctTorso" in the folder. Load it with your app and find a good iso-value to clearly show inner organs, e.g. the kidneys. What is the iso-level? Make a screenshot.

- **hint**: for better performance test your app in release mode with debugging off (standard hot key is Strg+F5 in Visual Studio)

Deliverable: The cpp file only! (For the optional task provide a screenshot, write the iso-value in file name)

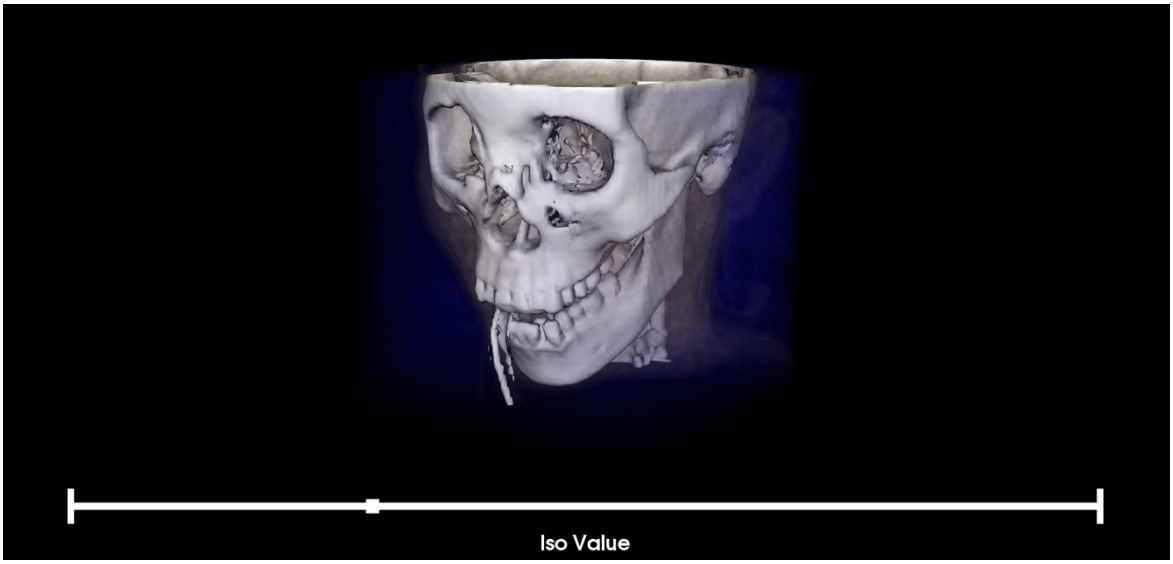

Figure 2-1. Combined rendering of ray-casted volume and isosurface.

Task 3: Compute Marching Squares by hand

Goal: Given a cell in a uniform grid, see the following picture, your task is to calculate the corresponding lines for the iso-value $v = -1$.

- Mark the locations where the isolines should intersect the edges. Remember that the values $v_{i,j}$ are interpolated linearly. Draw as exact as possible.

The next step is to connect the intersection points on the edges to create the isoline segments for this cell. You will discover an ambiguity that should be resolved explicitly by calculating the asymptotes.

- Write down the equation for the isoline and solve it for x and y respectively. You will find a singularity in each formula at the x/y-position where the asymptote orthogonal to the x/y-axis intersects. Draw the asymptotes into the given picture.

Resolve the ambiguity with the help of the asymptotic decider you know from the lecture.

- Finally draw the isoline segments for $v = -1$