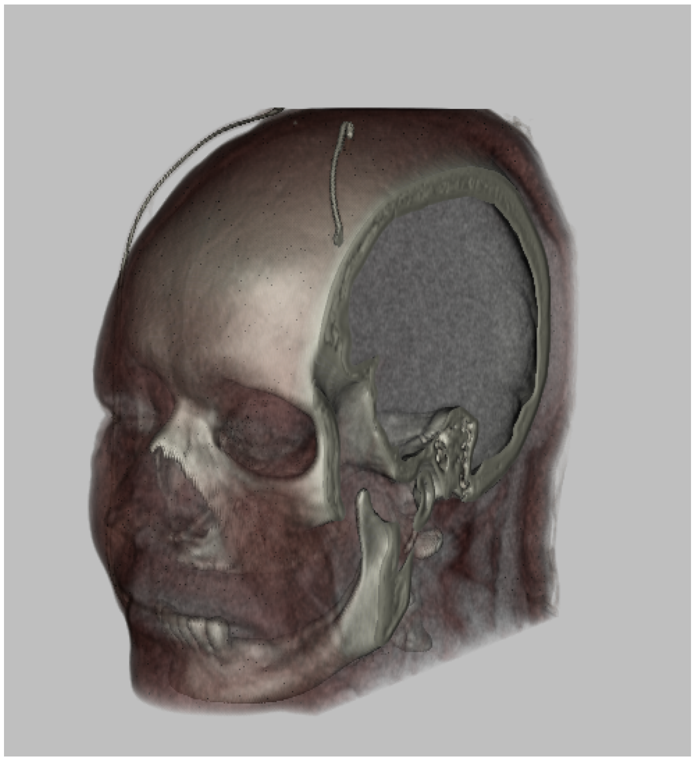
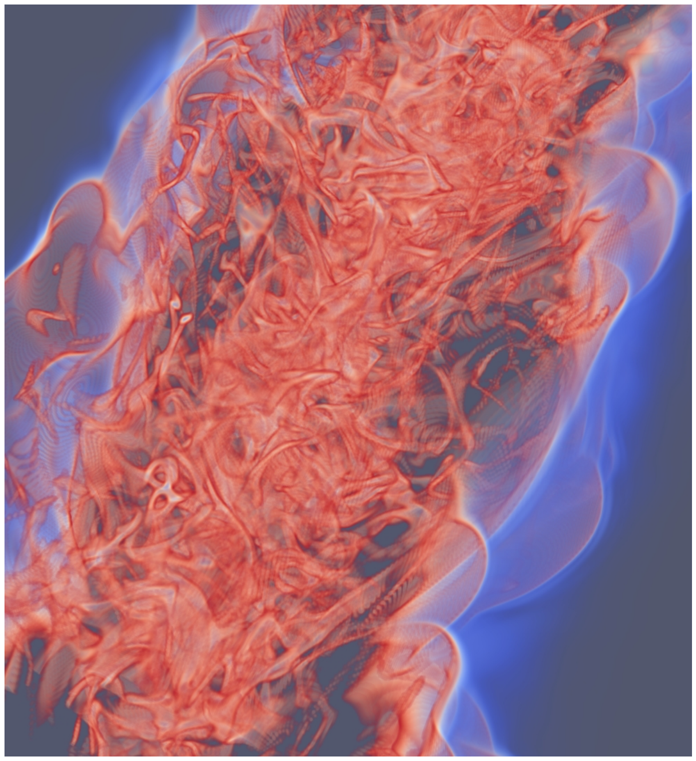
CS53000 - Spring 2021

Programming Assignment 3 - Volume Rendering

Key Dates

Handed out: March 1, 2021
Due: March 15, 2021 at 11:59:59 pm

Objective

The topic of this assignment is the visualization of 3D scalar fields through direct volume rendering. You will experiment with transfer function design and revisit some of the tasks of the second assignment to compare the effectiveness of isosurfacing and volume rendering in two application scenarios. Specifically, you will apply volume rendering to a medical dataset (similar to the CT volume used in the previous assignment) and to a computational fluid dynamics (CFD) simulation of turbulent combustion.

Background

The key to achieving good results with volume rendering is to select an effective transfer function. We saw in class that a good transfer function should reveal boundaries present in the volume, when such boundaries exist. When clear boundaries are not present, the transfer function should be designed to reveal important geometric structures in the data. In this assignment, you will be working with two datasets that illustrate these two scenarios: the CT dataset contains boundaries corresponding to the interface between different tissue types while the CFD dataset describes the spatial distribution of vorticity in a turbulent combustion and is globally smooth. The project starts by asking you to identify remarkable (iso)values in each dataset, which you will then use as reference points to create your transfer function. The third part of the project invites you to compare the respective pros and cons of volume rendering and isosurfacing in the context of these two datasets.

Tasks

Task 1: Important Isosurfaces

Your first task consists in determining for each dataset a set of isovalues that capture *salient* (remarkable) structures in the considered field. To do so, you will use the code that you wrote for the second assignment to identify important isosurfaces. In the case of the head dataset, salient isosurfaces capture boundaries corresponding to the skin, muscles, skull (and teeth). In the case of the combustion dataset, you will look for isosurfaces that reveal the sheet and tubular structures that are present in the flame. Bear in mind that fuzzy structures might be poorly captured by isovalues. Using different opacities for different isosurfaces, create for each dataset a visualization showing all isosurfaces simultaneously.

Deliverables: Create two executables for this task: `salient_head.py` and `salient_flame.py` that each contain the (*hardcoded*) information needed to visualize the salient isosurfaces of the corresponding dataset using transparency. In both cases, your executable will obtain the name of the file to visualize from the command line (which allows us to specify an arbitrary location and resolution for the input file).

```
> python salient_head.py <head.vti>
> python salient_flame.py <flame.vti>
```

Report: Describe in the report how you selected the isovalues for each dataset. Include pictures showing *each isosurface individually* and other images showing *all isosurfaces combined* using transparency. Make sure to use the **same camera setting** across all images corresponding to the same dataset. To that end, refer to the code sample that showed you how to print out the current camera setting during an interactive session as well as save the current frame to file. Once you identify a suitable camera position, simply hard code the corresponding parameters in your program or create a mechanism to import those settings from file on the command line.

Task 2: Transfer Function Design

Now that you have found good isovalues, you will design a transfer function for each dataset based upon those values. For that, create a vtkVolumeProperty by following the example provided in `Examples/VolumeRendering/Python/SimpleRayCast.py` to define both color and opacity transfer functions in order to emphasize the selected isovalues. You are already familiar with vtkColorTransferFunctions from previous assignments. The opacity transfer function is defined through a vtkPiecewiseFunction. Your objective in designing the opacity transfer function is to reveal as much as possible of the internal structures of each dataset. The volume rendering itself will be performed by raycasting using a vtkSmartVolumeMapper. Note that this implementation will automatically determine the hardware resources available and perform GPU-based raycasting whenever possible. Select the compositing blend mode of `vtkSmartVolumeMapper` for value compositing along each ray. Your implementation should produce high-quality renderings by combining trilinear interpolation and small sampling distance along each ray: you will select `SetInterpolationTypeToLinear()` in the API of vtkVolumeProperty and manipulate the discretization along each ray via `SetSampleDistance()`. You will need to experiment with different values of the sampling distance to determine the precision necessary to obtain good results. Good results in particular should not exhibit aliasing artifacts such as moiré effect. Note that an appropriate value of the sampling distance depends both on the smoothness of the data and on the properties of your transfer function. Finally, you should activate the shading option (via `ShadeOn()` in vtkVolumeProperty()) in your rendering to further improve the visual quality of your results.

Deliverables: create two executables `dvr_head.py` and `dvr_flame.py` that contain the information necessary to create high quality renderings of the corresponding dataset. In particular, the opacity and color transfer functions must be hard coded in these programs. Like for the first task, your executables must obtain the name of the data file from the command line.

```
> python dvr_head.py <head.vti>
> python dvr_flame.py <flame.vti>
```

Report: Provide a detailed description (including diagram) of the various transfer functions you created along with a justification of the choices made. What method did you use to create each opacity transfer function? What do you consider to be the strengths and limitations of your solutions? Include in the report several images for each dataset that highlight the effectiveness of your transfer function and the quality of your volume rendering parameters.

Task 3: Volume Rendering vs. Isosurfacing

Provide for each dataset a side by side comparison between the results obtained for isosurfacing (Task 1) and volume rendering (Task 2). Make sure to **use the same camera settings** for both techniques in order to facilitate their comparison. No additional code should be written for this task: simply use the executables created for the previous tasks to create the images that will be included in the report.

Report: Comment on the differences between the two techniques. Illustrate your argumentation by zooming on particular features of each volume. For each dataset, which technique do you find most effective? Why? Be specific.

Summary Analysis

Include in the report your critical assessment of volume rendering: What are in your opinion the pros and cons of this technique? Refer to the tasks of this project to justify your opinion.

Data Sets

You will be visualizing two datasets for this assignment. The first one is a head CT scan dataset similar to what you used for Project 2. Here, the data corresponds to a male subject, part of the Visible Human Project (National Library of Medicine). The second dataset corresponds to a computational fluid dynamics simulation of a turbulent combustion. In this case the provided scalar volume corresponds to the magnitude of the vorticity field, a quantity that is derived from the flow velocity and can be used to identify vortices in numerical datasets.

Note that both datasets have been low-pass filtered (“smoothed”) to facilitate their visualization (*e.g.*, reduce aliasing issues).

The datasets are available online. All datasets are of type `vtkStructuredPoints`.

- **Visible Male Head:**
  - CT scan, low resolution (`unsigned short`, 12.2MB)
  - CT scan, high-resolution (`unsigned short`, 98MB)
- **CFD Dataset:**
  - Vorticity magnitude (low-resolution) (`unsigned short`, 2.6MB)
  - Vorticity magnitude (high-resolution) (`unsigned short`, 18.5MB)

Submission

Submit your solution for this project on Blackboard before **March 15, 2021 at 11:59:59 pm**. Refer to the instructions below.

- Include all program files (`salient_head.py`, `salient_flame.py`, `dvr_head.py`, `dvr_flame.py`) along with any other source code you may have.

- Include **high resolution sample images** showing results for each task.
- Include a **report** briefly summarizing what you have done and answering all the questions asked. As always, the report should include high-resolution images.
- Include **`README.txt`** file with execution instructions (optional).
- Include all submitted files in a single directory named **`<myLogin>_p3`**, where **`<myLogin>`** is your Purdue login.
- **Do not** include binary file
- **Do not** include data files
- **Do not** use absolute paths in your code

Last modified Sun Feb 28 2021