Prof. Dr. Thomas Schultz
Mohammad Khatami (khatami@cs.uni-bonn.de)
Ikram Jumakulyyev (ijumakulyyev@cs.uni-bonn.de)

University of Bonn
B-IT/Computer Science II
June 26, 2020

Summer term 2020

Visual Data Analysis

Assignment Sheet 10

Solution has to be uploaded by July 6, 2020, 8:00 a.m.
to `https://uni-bonn.sciebo.de/s/eCj5G5qZVJkWd9M` with password `vda.2020`

Please bundle the results (as PDF) and scripts (*.py/*.ipynb files) in a single ZIP file. Submit each solution only once, but include names and email addresses of all team members in the PDF and each script. Name the file `vda-2020-xx-names.zip`, where `xx` is the assignment sheet number, and `names` are your last names.

If you have questions concerning the exercises, please write to our mailing list:
vl-scivis@lists.iai.uni-bonn.de.

Exercise 1 (Illustrating Marching Cubes Cases, *12 Points*)

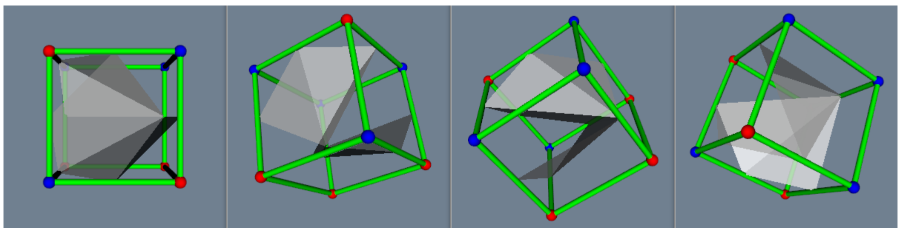

Figure 1: Differently rotated illustrations of a complex triangulation for Marching Cubes Case 12.

It is difficult to visualize some of the more complicated Marching Cubes cases in single two-dimensional projections. In this assignment, you will therefore create an interactive visualization in which the cube can be freely rotated to gain a better understanding, as demonstrated in Figure 1.

On the lecture homepage, we provide a framework `MarchingCube.py` for illustrating the different base cases, which can be selected via the `-c` command line parameter. For each case, a single voxel is rendered with colored vertices indicating the sign pattern of the corresponding base case. However, the correct triangulation is only implemented for Case 2.

a) Your first task is to implement one valid triangulation for Case 12. We recommend referring to `getPos` to understand how the given code indexes the vertices, and to `PointColoring` to see which vertices have different signs in Case 12. The example code in `case2` should help you understand how you can add triangles. Once you have a visualization, rotate it to convince yourself that the surface correctly separates the differently colored corners of the cube. (6P)

b) Case 12 is one of the ambiguous cases. In fact, the asymptotic decider is required on two faces, leading to four possible surface configurations overall. Implement the three remaining subcases of Case 12, and allow the user to switch between them using the `-s` (subcase) command line flag. Visually verify in all cases that you have a correct solution and submit screenshots of all cases along with your code. (6P)

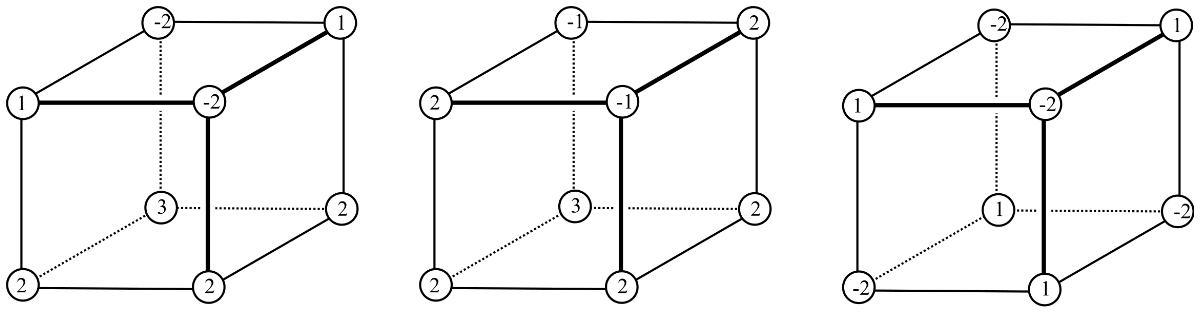

Figure 2: Your task is to apply Marching Cubes to these cells.

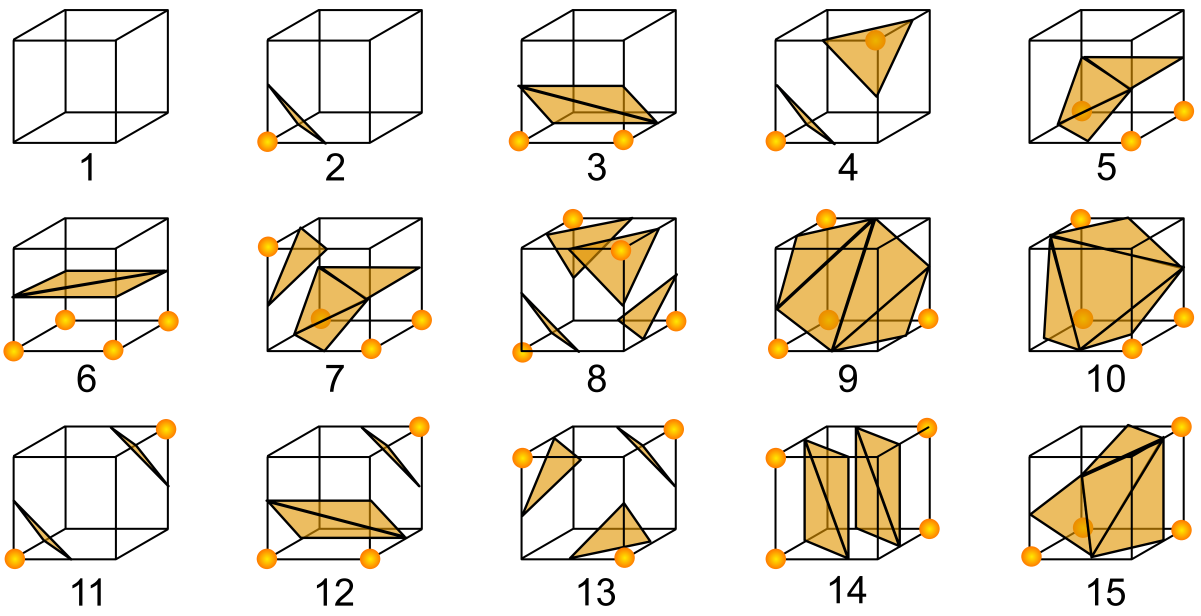

Figure 3: Marching Cubes cases. Please remember that, depending on the outcome of the decider, some cases might result in different triangles than shown in this table.

Hint: Even though it is possible to figure out the different triangulations yourself, you are likely to find it time consuming and tedious. We highly recommend using the triangulations suggested in `nielson-asymptotic-decider-1991.pdf`

Exercise 2 (Marching Cubes, *9 Points*)

Given the cells in Figure 2, apply the Marching Cubes algorithm for the isovalue $c = 0$ by drawing the resulting triangles of the isosurface. Indicate which of the 15 base cases shown in Figure 3 has to be used to connect the edge intersections in each case. Mark the faces of the cells that result in ambiguous configurations and apply the asymptotic decider where needed.

Exercise 3 (Asymptotic Decider, *6+3 Bonus Points*)

a) According to Chapter 9, the direct way to evaluate the asymptotic decider involves re-writing the standard equation for bilinear interpolation into a form that allows us to read off the intersection

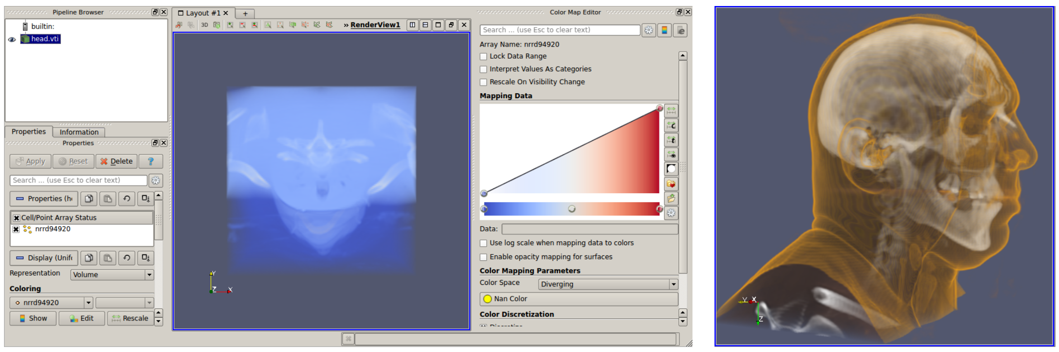

Figure 4: Initial direct volume rendering in ParaView (left) and result after adapting the transfer function and rotation (right)

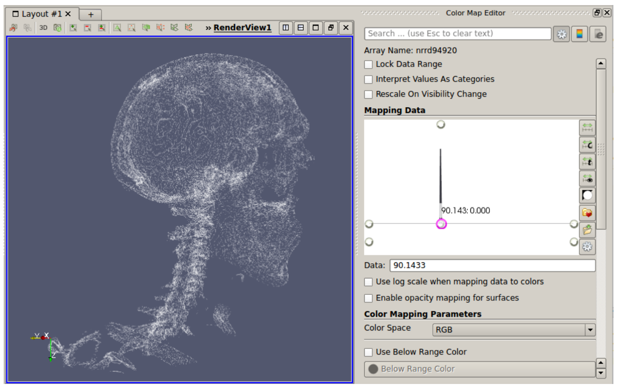

Figure 5: Result from a transfer function that is close to a delta peak.

point of the asymptotes and the function value at that point. Specify explicit formulas to compute η, x_0, y_0, and γ from $f_{i,j}$, $f_{i+1,j}$, $f_{i,j+1}$, and $f_{i+1,j+1}$. (4P)

b) In cases which require the asymptotic decider, do we ever have to divide by zero in the equations you derived in a)? Why? (2P)

c) Unfortunately, the scientific paper that introduced the asymptotic decider contained a serious typo. You can download this paper from our homepage. We will give you a bonus point if you identify the equation in that paper where the equivalent of γ (in our notation) is computed, and two more if you tell us what is wrong with this equation, and how to fix it.

Exercise 4 (Direct Volume Rendering in ParaView, *10 Points*)

In this task, you are going to learn how to perform direct volume rendering in ParaView.

a) Open ParaView and load the `head.vti` dataset. In the "Properties" tab, hit "Apply" to actually make it available. Now, change the representation to "Volume" and confirm your choice. At this point, you should see a result as in Figure 4 left. Now, modify the transfer function and rotate the volume so that you obtain a result similar to Figure 4 right. Please submit a screenshot of your result, including the transfer function editor. (3P)

b) In Figure 5, a transfer function has been chosen that assigns full opacity, but only in a very narrow value range (close to a delta peak). What might be the intended effect of such a transfer function?

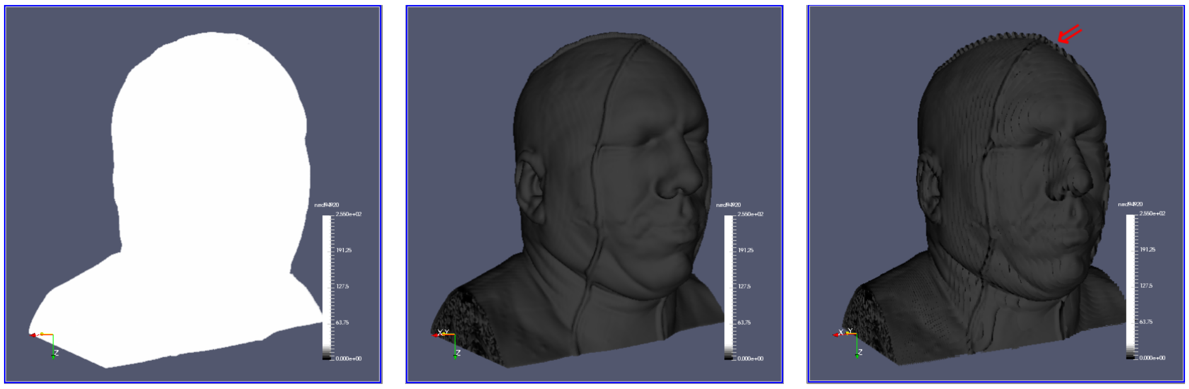
Figure 6: Silhouette rendering (left), same surface with details (center), and artifacts during interaction (right).

Why does it not lead to cleaner image? Which method do you remember from the lecture that could improve the rendering in such a case? (3P)

c) Figure 6 left shows the result of using a step function as a transfer function, which visualizes the silhouette of a surface. Provide a screenshot that additionally reveals the surface shape, as in the middle image. What was missing in the first case, and had to be added for the second one? (2P)

d) While rotating the volume, you will notice artifacts such as the ones highlighted by the red arrow in Figure 6 right. They vanish automatically once you release the mouse. Briefly explain this effect. (2P)

Exercise 5 (Two-Dimensional Transfer Functions, *6+14 Bonus Points*)

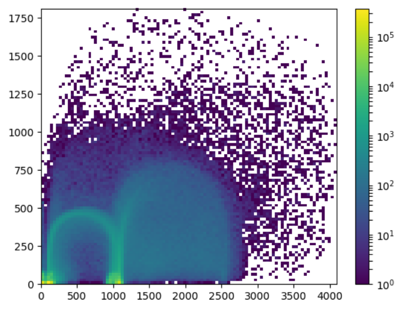

Figure 7: 2D Histogram of the volume `headsq.vti`.

In Direct Volume Rendering, 2D Transfer Functions map intensity and gradient magnitude at each point of a scalar volume to color and opacity. Their design is guided by 2D histograms, such as the one for the input volume `headsq.vti` which is shown in Figure 7. Please answer the following two questions about it:

a) Based on the 2D histogram in Figure 7), how many materials can you distinguish in this volume? State a number and briefly justify it, possibly with a sketch. (2P)

b) Use isosurfaces (for example, in ParaView) to find out what the materials you've identified in the 2D histogram represent. Name the materials, and submit screenshots for justification. (4P)

To obtain up to 14 bonus points, you may also submit your Python code along with explanations and screenshots for the following tasks:

a) Compute the gradient magnitude of the dataset `headsq.vti` that is available from the lecture homepage. To this end, extend the framework in `directVolumeVisualization.py` by implementing the function `computeGradientMagnitude`, which should return a vtkImageData object containing the gradient magnitude at each voxel. (5P)

   Hint: You are welcome to use the Sobel filter provided in VTK. Alternatively, it only requires a few lines of code to implement some numerical scheme for taking derivatives yourself (remember Chapter 8).

b) Finish the function "visualizeHistogram(...)" to plot a 2D histogram of data and gradient magnitude. The result should look similar to Figure 7. (4P)

   Hint: If you use a suitable package such as matplotlib, solving this part requires very little code.

c) Use this information to modify the transfer function, which is defined by `volumeScalarOpacity`, `volumeGradientOpacity` and `color` in `volumeVisualization`. Please submit your code and a screenshot that uses direct volume rendering to show multiple material boundaries semitransparently, in different colors. (5P)

Exercise 6 (Style Transfer Functions, *7 Points*)

The transfer functions we discussed in the lecture assign different colors to different materials in the volume. The paper `bruckner-style-transfer-functions-2007.pdf` that is available from the lecture webpage achieves quite impressive visual effects by assigning a more complex rendering style instead of a single color. Please read it and answer the following questions in your own words. Remember that **we will not grant even partial credit for copy-pasted text.**

a) In Equation (2) in the paper, why does it make sense to multiply α_d and α_u? Briefly describe the qualitative effect of the exponent that is applied to α_u. (2P)

b) The procedure for style transfer function lookup, which is explained in Section 4.1 and illustrated in Figure 8, is relatively complex. What would be a simpler alternative? Why did the authors invent the more complex procedure that is described in the paper? (2P)

c) In which case could we skip the index function texture lookup? Why? (2P)

d) Even though style transfer functions greatly increase flexibility compared to conventional color transfer functions, Table 1 in the paper reports that they are even slightly faster in two specific scenarios, namely Figures 7 (a) and (d). How do the authors explain this? How do these two cases differ from Figures 7 (b) and (c), where style transfer functions are slower than regular transfer functions? (2P)

Good Luck!