Assignment 1 Visualization

D. J. van den Brand - 0772180

December 12, 2016

1 Data sets

This section elaborates on all the interesting features in the provided data sets. The Images and parameters are located in the appendix.

1.1 Orange

The slicer technique can be used to look for artifacts where the location is already known. As seen in Figure 3a there are pits in the orange. Because the shape of the pits do not really matter and we already know where they should be it is easy to recognize these with a slicer that leaves out details.

In Figure 3b a Maximum intensity projection is used. The transfer function is set such that only the outer layer of the orange is visible. It is visible that the orange is either a bit weirdly shaped or is pressed between two plates which are clipped from the data set.

And in Figure 3c this same projection is used with a more complex transfer function that can be seen in Table 1. Because composting takes the opacity of every voxel into account it will show more inner details. But is does not account for depth so the real shape is not seen properly.

In Figure 3d the shape is clearly visible. Because the composite technique makes sets the opacity according to all points along the sampled ray it gives a better insight in what the shape is. Also the shading is proportional to the gradient of the surface. This makes the inner structure more clear. And finally the shading also has a factor depending on the distance the ray has traveled, which gives the perception of depth.

1.2 Pig

The pig data set has a couple of nice features. The slicer technique as seen in Figure 4a can best be used to see the wholes in the top and bottom of the pig. In Figure 4b it is clearly visible that the wood where the pig stands on has two different densities. These have been colored by different values in the transfer function in Table 2. Also due to the shading it is visible that there are flowers on the outside which stand out due to the shading model. In Figure 4c it is clearly seen that there are some coins inside the pig. Also it is visible that the pig consists of a double layer. When the 2d transfer function is focused on the least intense values the measuring artifacts become visible, as seen in Figure 4d. This is probably just noise but it is remarkably structured. For narrow values in the 2d transfer function many artifacts arise. Increasing the sampling rate does not fix this problem, as seen in Figure 4e and 4f. In Figure 4g and 4h it is visible that shading brings a feeling of depth to the image. If the shading is turned off it is much harder to see that there are separate coins instead of a single clump of material.

1.3 Tomato

The slicer gives a good contrast between different parts of the tomato in Figure 5a. There are some very light artifacts at the edge of the tomato. This could be a damage in the outer layer that makes it softer. This area is colored green in Figure 5b. However the tomato data set is harder to visualize with a volumetric rendering.

Because the tomato is very watery it has a very uniform density and does not have large gradients. Therefor the transfer function will not give very large differences between the gradients, as seen in Figure 5c.

1.4 Other sets

The bonsai tree does not have many special features. It is clearly seen in Figure 6 that it has a root and leaves.

2 Implementation

In the project 4 different classes are used for ray casting. The CenterSlicer implementation was already provided. Below the other implementations are discussed. All these implementations make use of the same ray casting approach as explained in the following section.

2.1 Ray casting

The user provides a minimal amount of steps s_0 that have to be taken through the data. If we divide the data equally into s_0 slices the distance between these slices should be at least $\mathrm{dV} = d_{min}/(s_0 + 1)$, where d_{min} is the smallest possible intersection. Based on the zooming of the camera an appropriate distance between each pixel is calculated by the framework. Then for each pixel a point q_0 is calculated for which the ray is cast, as seen in Figure 1. By intersecting the ray with each face of the bounding box yields two intersection points $q_0 - \lambda_0 \mathbf{v}_{\text{view}}$ and $q_0 + \lambda_1 \mathbf{v}_{\text{view}}$. So by sampling each ray from $q_0 - \mathrm{dV}\lceil\lambda_0/\mathrm{dV}\rceil \cdot \mathbf{v}_{\text{view}}$ to $q_0 + \mathrm{dV}\lceil\lambda_1/\mathrm{dV}\rceil \cdot \mathbf{v}_{\text{view}}$ we can calculate the value for each pixel.

There are three options implemented in how the sampled points are used to get a intensity value from the original data:

- Flooring. Gives the closest coordinate by simply rounding the coordinate system down. Because dV will mostly be greater then the spacing between data points a lot of data points will not be read using this method. This leads to artifacts because smooth transitions in intensity will have non regular gaps in between depending on the viewing vector.
- Nearest neighbor. Because the coordinate system aligns with the data this is simply implemented by component wise rounding. The results are almost the same as flooring but the outer slices of the data set are displayed more correctly. Given the small computational overhead this should not be used.
- Trilinear interpolation. By applying linear interpolation three times the intensity value is estimated using the 8 surrounding voxels. If the step size is sufficiently large this will smooth out the artifacts from flooring the values. However the computational overhead is quite substantial because on each retrieved sample point in the ray this interpolation has to be performed.

Then there are the following methods in how the actual color of the pixel is calculated.

2.1.1 Mip

In the maximum intensity projection only the voxel with the highest intensity along the sampled points is considered. So the alpha and color value are looked up from the transfer function and are displayed directly. This method will not merge data of all slices in each pixel but only from a single sample point per ray. Because the sampling points do not align with the data points a rotation or translation could shift the position of maximum. Therefor this method introduces a lot of noise when interacting with the data.

Also because not all slices are taken into account in each pixel a shape is hard to recognize. A good example of this can be seen in Figure

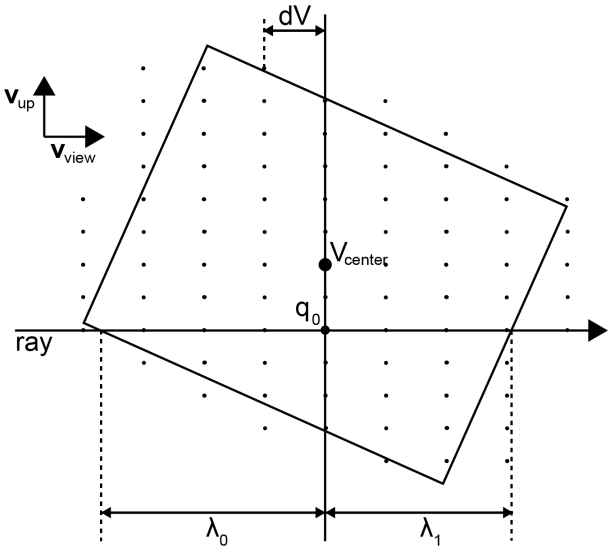


Figure 1: Intersection of a ray cast

2.1.2 Compositing

In the compositing projection all intermediate sampling points are computed from front to back according to the article. However when the sampling density changes there are less samples behind a certain sampling point. This is corrected by adjusting α to α' for each pixel by:

$$\alpha' \leftarrow \alpha(1 - (1 - \alpha)^{steps/300})$$

where 300 is a fixed reference value for the opacity the transfer functions are adjusted to.

2.2 Transfer function

A linear histogram is difficult to read when there are many empty points. Therefor a logarithmic histogram is implemented to have a better insight in how the data looks at a certain intensity.

Also to keep track off all the changes that are made to the transfer function the program now prints a LaTeX table and piece of source code with the new state after each change.

High opacity values lead to very sharp and crispy corners. And changes in low opacity values can have a big impact on the end result when they are in a sparse area. Therefore the control points of the transfer function are quadratically scaled to make it more sensitive to lower opacity values.

2.3 Shading

The simplified shading model as defined in the lecture slides is implemented. The parameters can be altered via the GUI, see Figure 2.

2.4 Responsiveness

To make sure the ray casting is quick enough to interact with there are a few optimization implemented when interactive mode is enabled:

- Shading is turned of
- When the ray has reached a cumulative density of greater then 0.7 sampling is prematurely stopped.

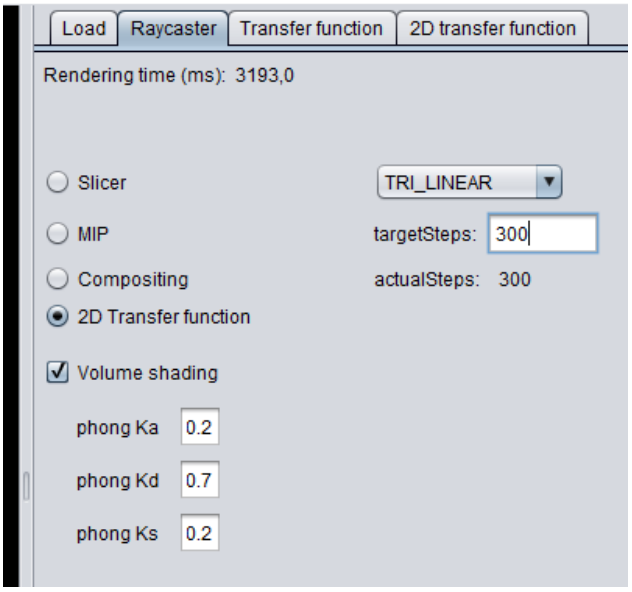
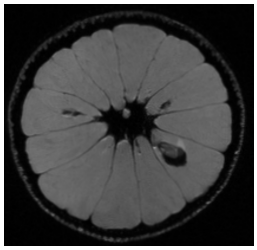


Figure 2: Ray caster panel

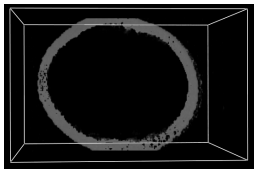
- The step size is adjusted to aim for a update rate of 15 FPS. This is done by calculating the worst render time per frame over the last 15 renders. Assuming the render time scales proportionally to the number of steps taken the new number of steps is set. On slow renders the lower limit is soon reached but on faster renderers this give quite a quality boost without loss of performance.
- The image width and height is scaled down by a factor 1.5 before doing to ray trace. Then the GPU is used to scale it up to the original size again on the screen.
- The voxels are never interpolated and always use the floor function.

3 Appendix

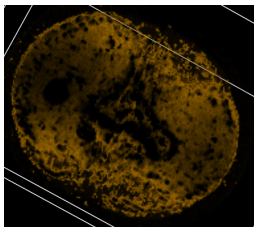
A Images



(a) Slicer

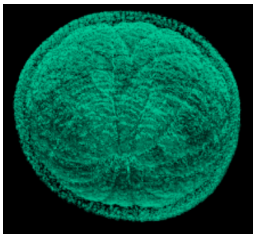


(b) MIP contour



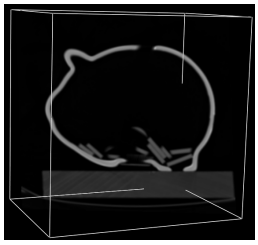
(c) MIP inner fruit

(d) Compositing

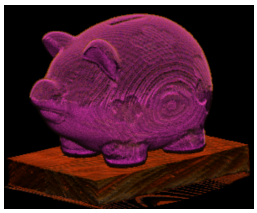


(e) 2D-tf

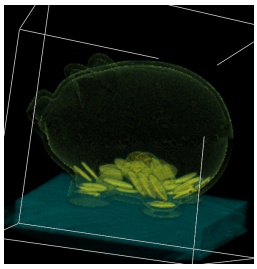
Figure 3: Orange screenshots



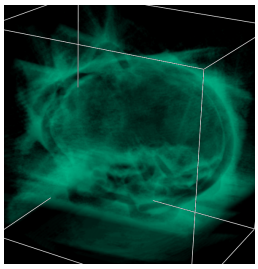
(a) Slicer



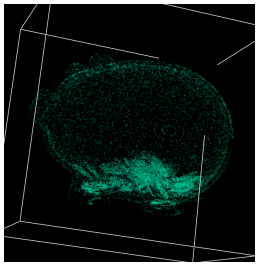
(b) Compositing outer



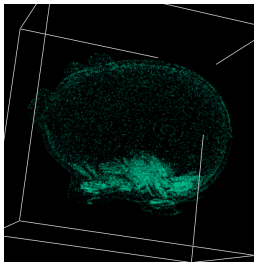
(c) Compositing inner



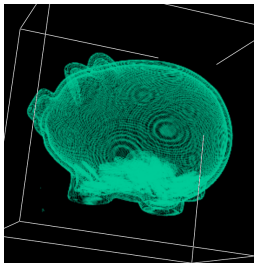
(d) Measurement errors



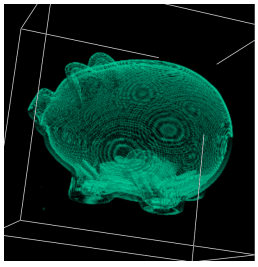
(e) 2D-tf 300 steps



(f) 2D-tf 2500 steps

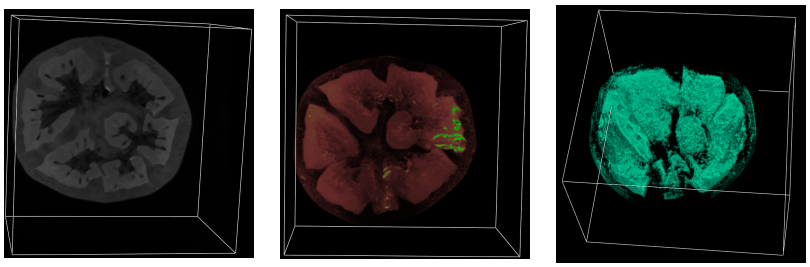


(g) 2D-tf no shading



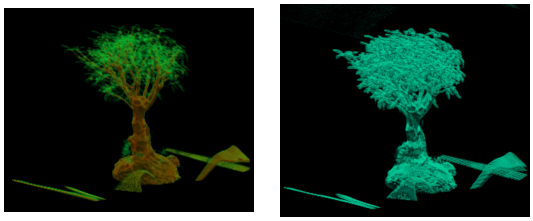
(h) 2D-tf with shading

Figure 4: Pig screenshots



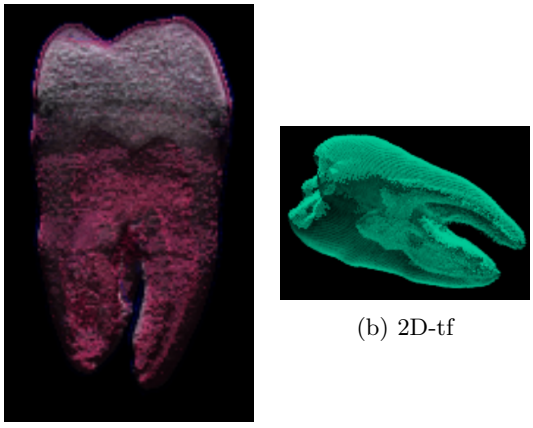
(a) Slicer (b) MIP (c) 2D-tf

Figure 5: Tomato screenshots



(a) Compositing (b) 2D-tf

Figure 6: Bonsai screenshots



(b) 2D-tf

(a) Compositing

Figure 7: Tooth screenshots

B Parameters

Value	Red	Green	Blue	α
0	0,00	0,00	0,00	0,00
21	0,00	0,00	0,00	0,00
38	0,25	0,25	0,25	0,07
41	1,00	0,40	0,00	0,27
43	0,45	0,30	0,00	0,10
72	0,48	0,32	0,00	0,36
78	1,00	0,67	0,00	0,76
103	0,00	0,00	0,00	0,41
205	0,00	0,00	0,00	0,00

Table 1: Transfer function Figure 3d

Value	Red	Green	Blue	α
0	0,50	0,50	0,50	0,00
34	0,50	0,50	0,50	0,00
36	0,58	0,43	0,38	0,02
38	0,80	0,20	0,00	0,41
46	0,00	0,00	0,00	0,01
48	0,40	0,20	0,00	0,26
53	0,50	0,50	0,50	0,00
79	0,50	0,50	0,50	0,00
85	1,00	1,00	0,00	0,12
94	0,50	0,50	0,50	0,00
104	0,55	0,47	0,53	0,13
119	0,60	0,44	0,56	0,00
121	0,93	0,24	0,76	0,11
134	0,98	0,21	0,79	0,21
152	1,00	0,20	0,80	0,11
172	0,61	0,61	0,59	0,00
255	1,00	1,00	1,00	0,00

Table 2: Transfer function Figure 4b

Value	Red	Green	Blue	α
0	0,50	0,50	0,50	0,00
19	0,00	0,00	0,00	0,00
52	0,00	0,40	0,40	0,06
54	0,04	0,41	0,41	0,00
77	0,50	0,50	0,50	0,00
82	1,00	1,00	0,00	0,11
88	0,50	0,50	0,50	0,00
255	1,00	1,00	1,00	0,00

Table 3: Transfer function Figure 4c

Value	Red	Green	Blue	α
0	0,50	0,50	0,50	0,00
23	0,79	0,02	0,02	0,00
36	1,00	0,40	0,40	0,16
46	1,00	0,40	0,40	0,45
59	1,00	0,40	0,40	0,63
94	0,00	0,80	0,00	1,00
223	0,06	0,79	0,06	0,38
224	0,68	0,68	0,68	0,00
225	1,00	1,00	1,00	0,00

Table 4: Transfer function Figure 5b

Value	Red	Green	Blue	α
0	0,00	0,00	0,00	0,00
24	0,00	1,00	0,20	0,00
30	0,00	1,00	0,40	0,04
34	0,08	0,90	0,00	0,04
36	0,00	1,00	0,20	0,00
38	0,18	0,84	0,15	0,04
41	0,45	0,60	0,07	0,05
45	0,41	0,41	0,00	0,49
51	0,53	0,20	0,00	0,16
135	0,47	0,20	0,00	0,00
142	0,47	0,20	0,00	0,00
148	0,46	0,20	0,00	0,00
155	0,44	0,20	0,00	0,00
169	0,20	0,20	0,00	0,40
253	0,00	0,00	0,00	0,39

Table 5: Transfer function Figure 6a

Value	Red	Green	Blue	α
0	0,00	0,00	0,00	0,00
440	0,00	0,00	0,00	0,00
494	0,00	0,00	1,00	0,01
519	0,00	0,00	1,00	0,03
541	1,00	1,00	1,00	0,00
673	1,00	1,00	1,00	0,00
711	0,81	0,53	0,62	0,12
746	0,60	0,00	0,20	0,13
777	0,68	0,19	0,35	0,32
786	0,72	0,29	0,43	0,00
812	0,79	0,48	0,59	0,01
846	0,88	0,70	0,76	0,01
871	1,00	1,00	1,00	0,00
922	0,88	0,88	0,98	0,00
1082	0,81	0,81	0,96	0,00
1167	0,93	0,93	0,99	0,03
1211	0,87	0,87	0,87	0,06
1265	0,92	0,92	0,92	0,01
1300	1,00	1,00	1,00	0,03

Table 6: Transfer function Figure 7a

Figure	Steps	Intensity	Radius	Opacity	Shading
3e	300	51	1.161	0.2	yes
4d	600	0	0.114	0.01	yes
4e	300	82	0.041	0.3	yes
4f	2500	82	0.041	0.3	yes
4g	300	77	0,447	0,3	no
4h	300	77	0,447	0,3	yes
5c	500	51	0.902	0.3	yes
6b	500	31	1.153	0.9	yes
7b	300	744	0.968	0.3	yes

Table 7: 2d transfer function parameter values