## Marching Cubes: Limitations

The marching cubes algorithm is easy to understand, but it has several limitations. First, it is not computationally optimal in either time or space, so it wastes CPU cycles and memory resources. It wastes memory and time by generating many examples of redundant output data, such as meaningless triangles of zero area sharing the same edge and vertex space as existing nonzero triangles. [1]

Second, isosurfaces produced by the marching cubes algorithm can sometimes be shaped badly so that they appear to have topological holes in regions that were smooth in the original object the data was sampled from, or other kinds of deviations that do not reflect the meaning of the data. This is related to how the algorithm resolves ambiguous cubes containing multiple disconnected regions of intersection with the isosurface. Since the marching cubes algorithm only directly observes edges of the polygonal isosurface geometry, it has to guess at the faces connecting the edges. For cases where multiple, disjointed sets of edges appear on the same cell, multiple ways of spreading facets between the edges are possible. The algorithm only works with one cell at a time, so it ignores information that could help it choose a good pattern of facets. The result is that some cells may contain an unexpected break in the surface, or join regions of the surface where they should be separate or vice versa (similar to the straits/isthmus problem in the 2D version, marching squares). [1]

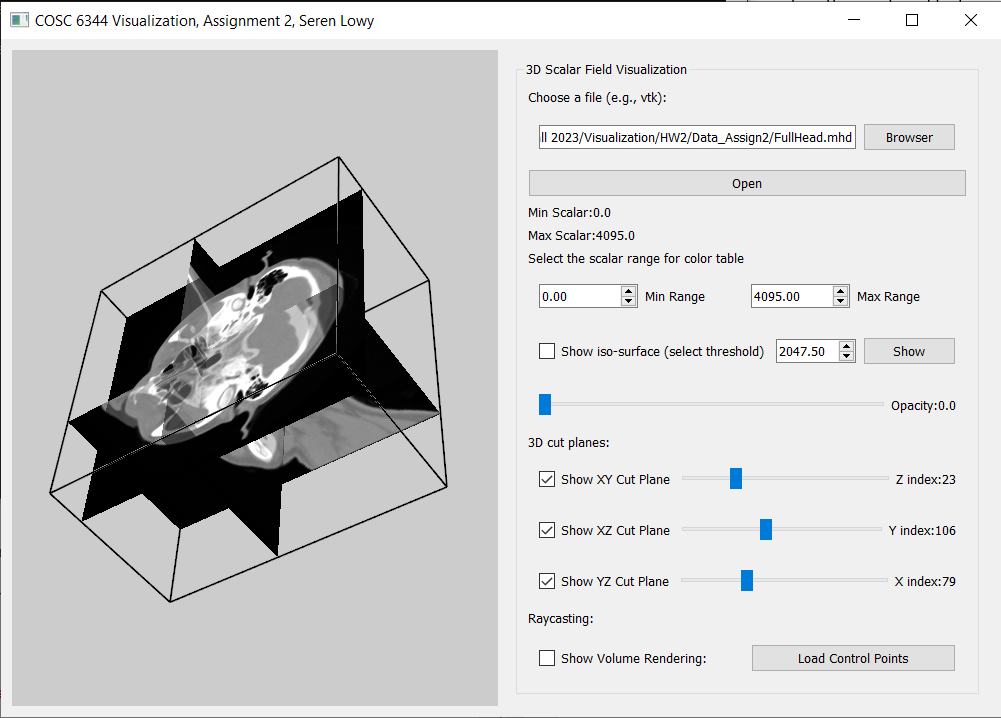
Finally, the marching cubes algorithm causes information loss by producing output artifacts. In marching cubes isosurfaces, small details in the data may be smoothed over, because a limited resolution of cells are used to capture the input, and features which occur and stabilize wholly within the space of a single cell are usually indistinguishable to the capture process of the algorithm. Sharp edge and corner-like features can be incorrectly captured as smooth or wide angles (“aliasing artifact”), even if they cover more than one cell. Gradient maps of isosurfaces that use marching cubes algorithm output can also be incorrect, in such a way that accidental spots of shadow or bright light can appear when the gradient map is visualized as a shading map, among other consequences. [1]

Sources: [1] Newman, T. and Yi, H. (2006). “A survey of the marching cubes algorithm.”

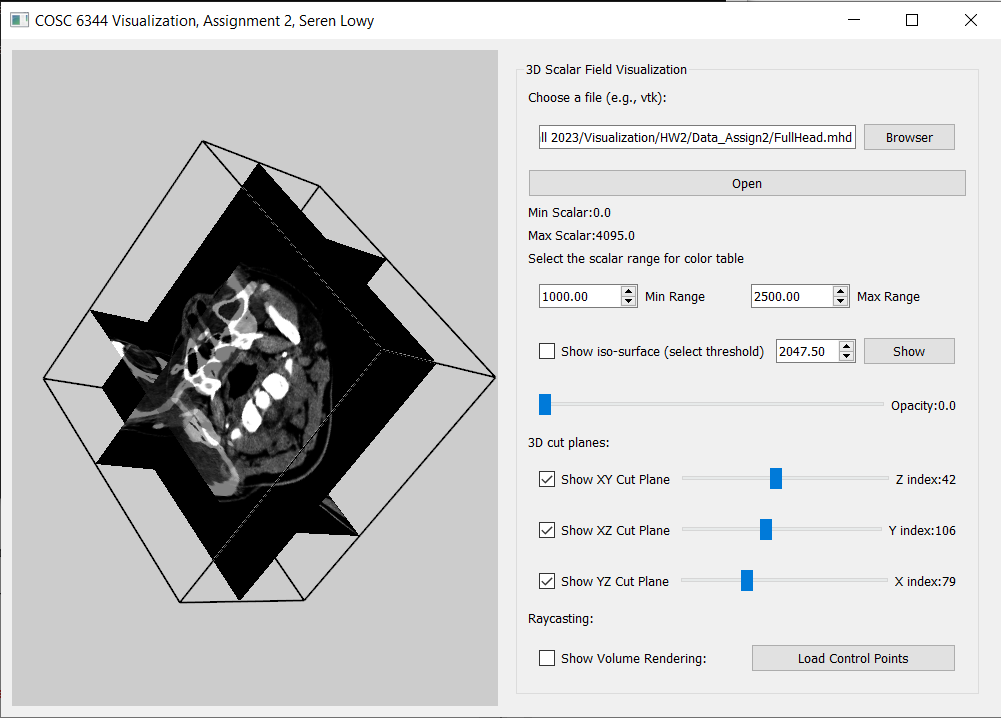
## Raycasting and Splatting: limitations

…

## 2. Cut Planes



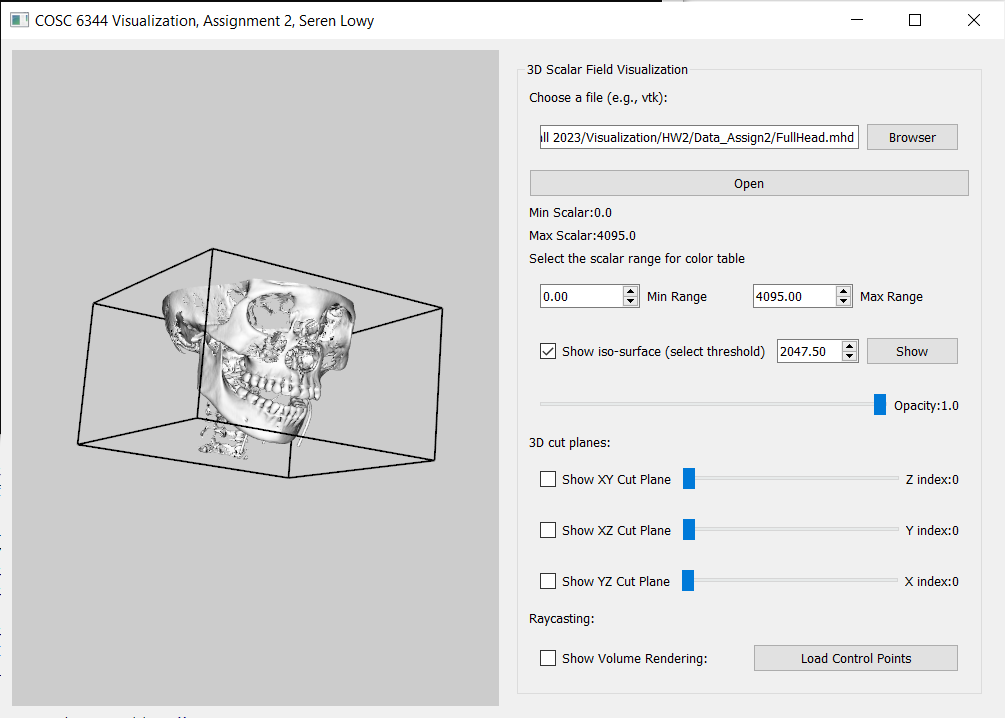
Above: screenshot of program window, showing 3 cut planes (YZ, XZ, XY), dataset “FullHead.mhd”, default values for dynamic scalar range.



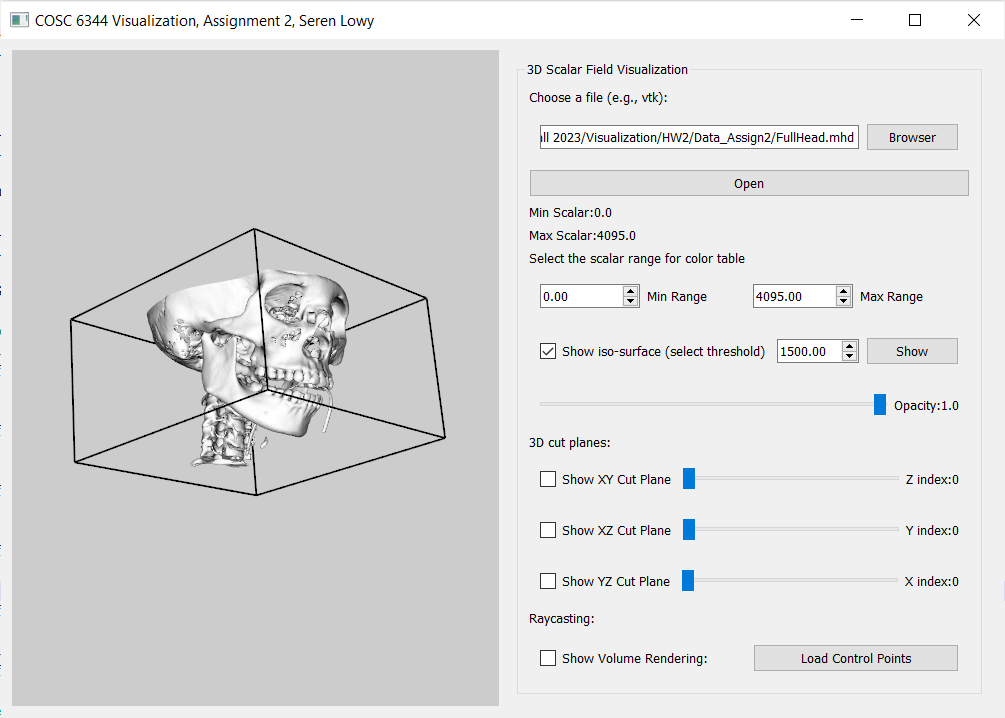
Screenshot of cut planes with narrower dynamic range. Some bone structures are more visible.

Please test my program UI to verify that checkboxes correctly show and hide the cut planes, Min Range and Max Range spinboxes interactively modify the colors visible in the cut planes, and Min range and Max range are not allowed to be in decreasing order (min > max, out of order).

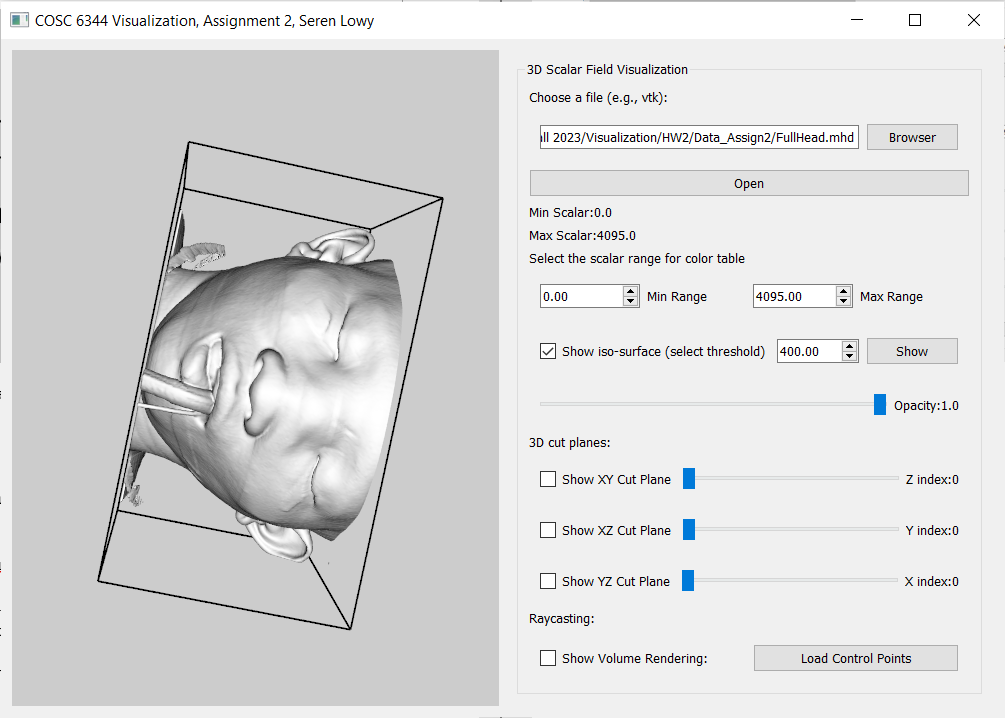
## 3. Iso-Surface Visualization



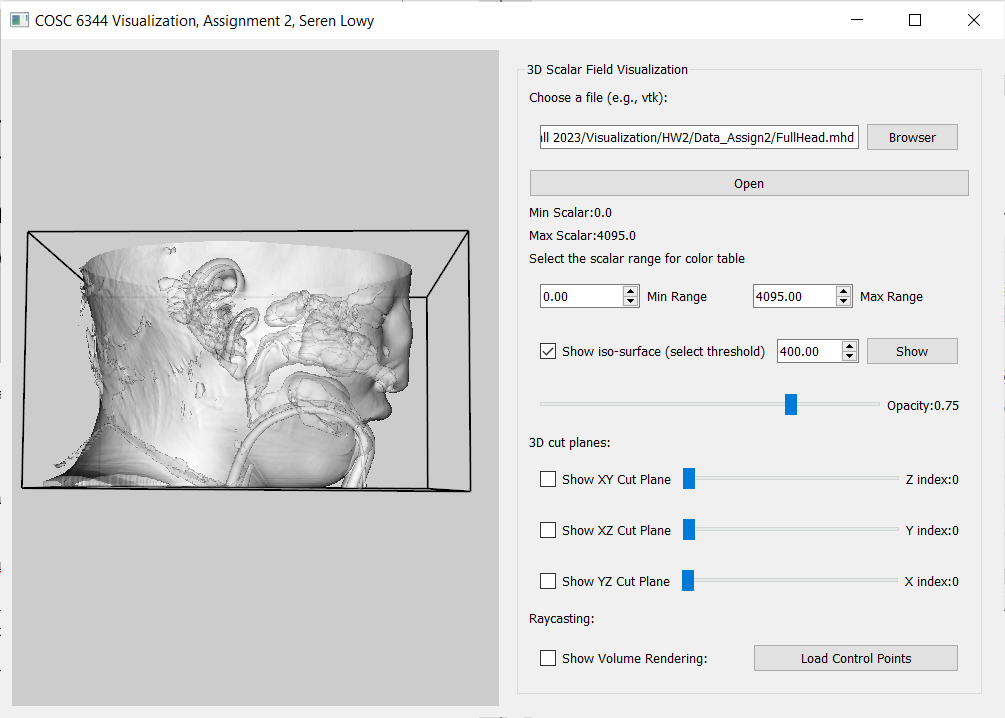
Above: The program window with an iso-surface of dataset “FullHead.mhd” at the default value (half the scalar range, 2047.50) and full opacity. The result looks like a skeleton skull face.



The iso-surface of 1500.00 value and full opacity, which looks like a skull with less of the bone worn out.

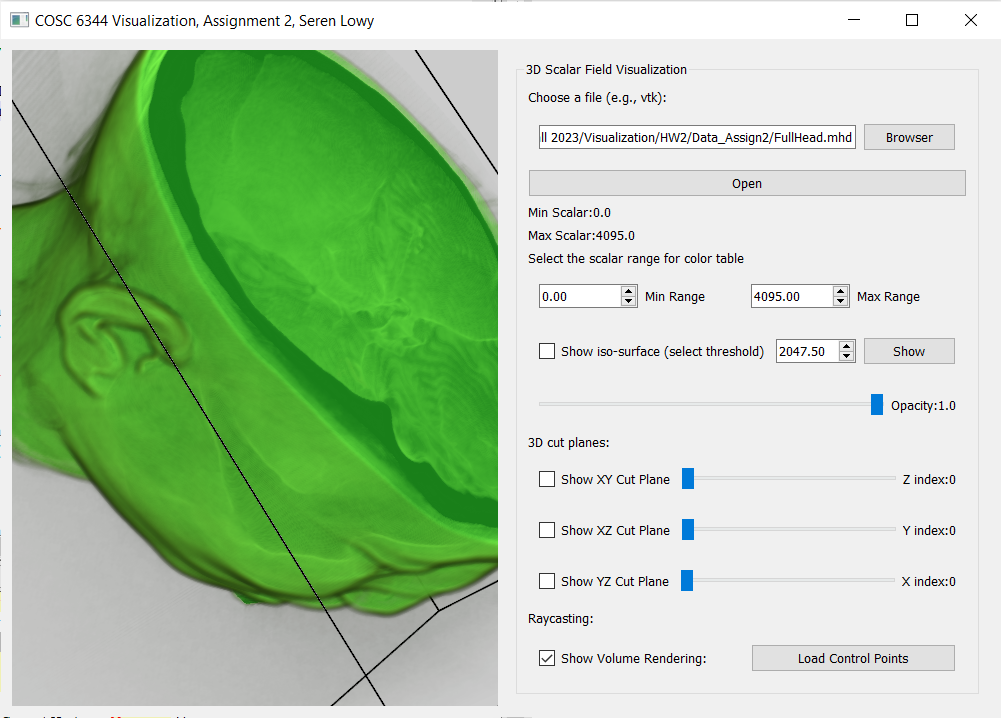


The iso-surface of value 400.00 and full opacity, which looks like an unconscious face of a hospital patient with a breathing tube.



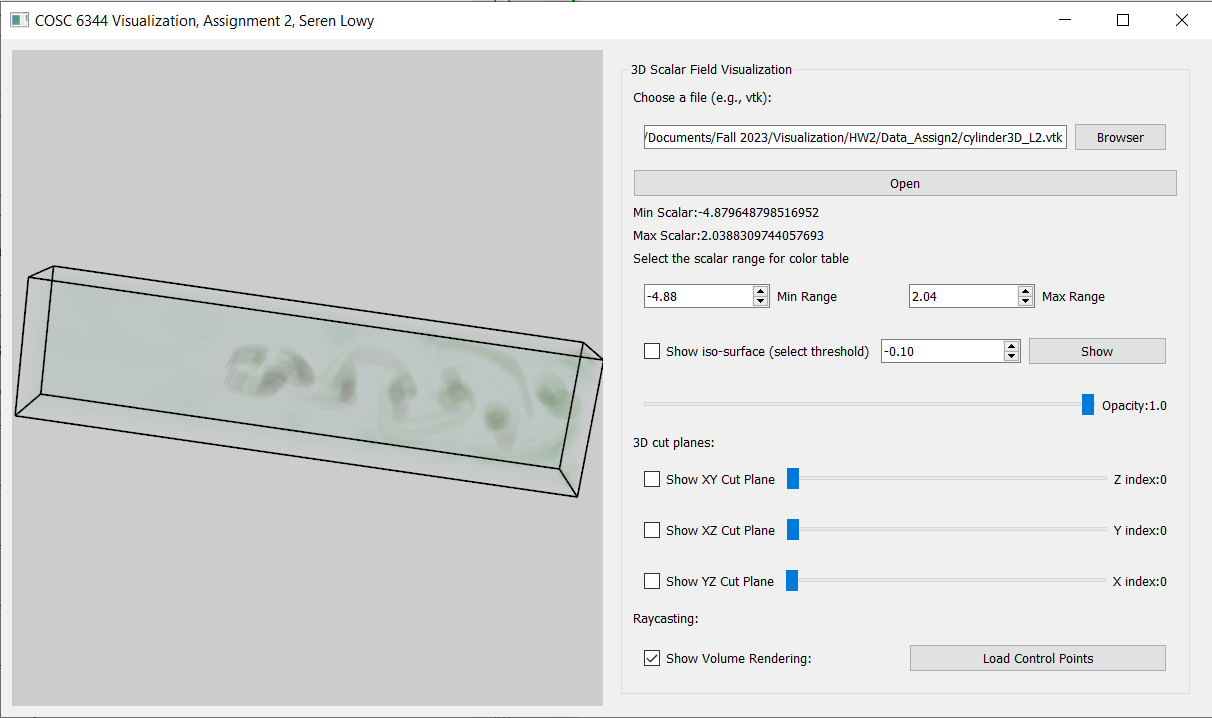
When opacity is changed to 0.75, we can see the sinus system and inner ears inside the patient’s face.

## 4.0 Raycast Volume Rendering

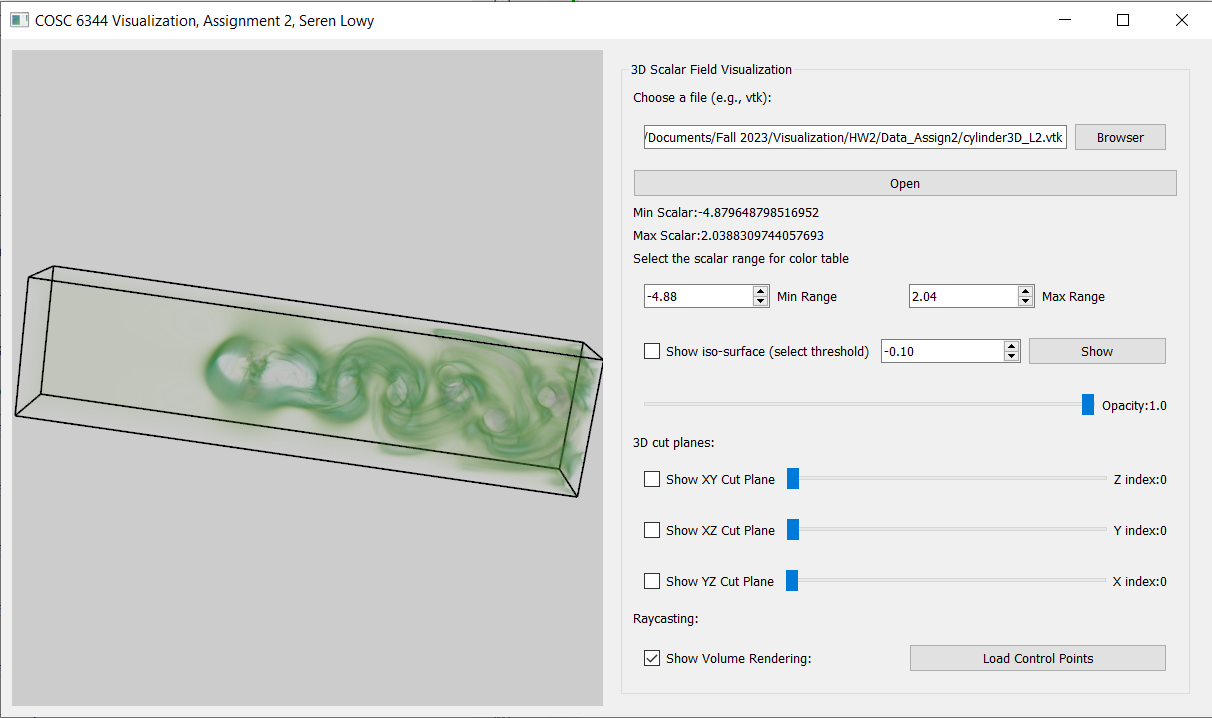


Above: Volume rendering of dataset “FullHead.mhd” in customized color. Regions of the patient’s brain are visible along with their face.

## 4.1 Raycast rendering, exploring Cylinder dataset



Above: Smaller tube-like structures (vortex cores)



Larger structures (flow regions surrounding the vortex cores)