Lab 4 — Scalar Data | CPSC 8810.001 Data Visualization

Name: Chaoren Li
username: chaorel

Data Reader

This part is done by a NDDR reader class. It basically reads in a nddr image data by a given path. It contains functions to interpolate and normalize data.

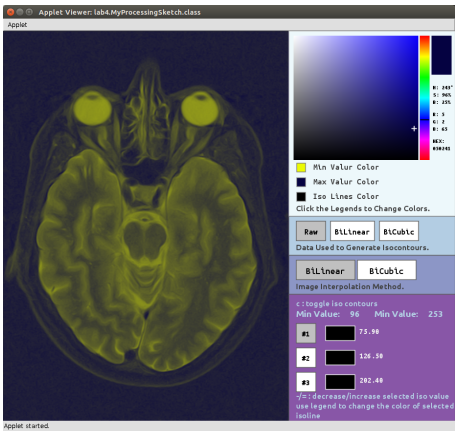
Color Mapping

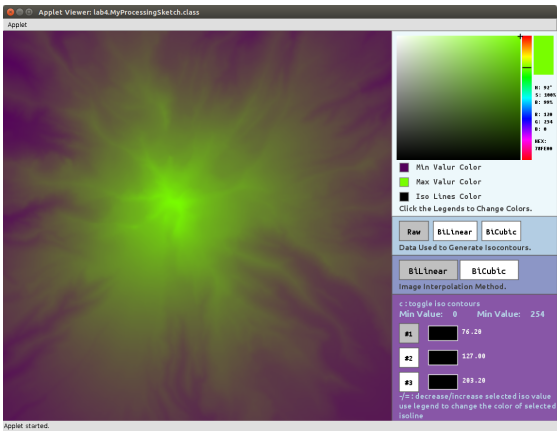
I tried several combinations of color, then I came up the idea to have a color picker. Instead of RGB style, I chose to use HSB style because as the idiom states, make it right in grayscale, then add color. I also put corresponding RGB and HEX values for convenience.

- For the test data, black and white is good enough to represent the data. One thing worth mentioning is this data ranges from 0 to 0xffff, so a modification is needed when mapping it.

- For the brain data, it contains more fine structures. First thing I did is invert the value, because the black values are we want explore. I used colors that call more visual attention, bright yellow for example, and for make it more popping out, I used dark blue as the opposite.

- The mtHood data is a little interesting, because the transition it contains is not as clear as the brain one. So I chose colors that opposite on the color wheel as well.

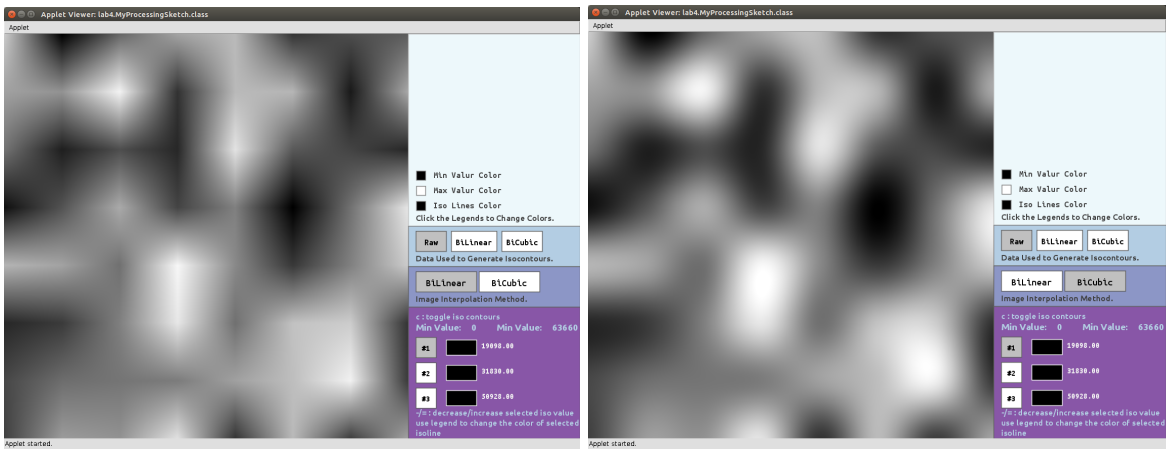




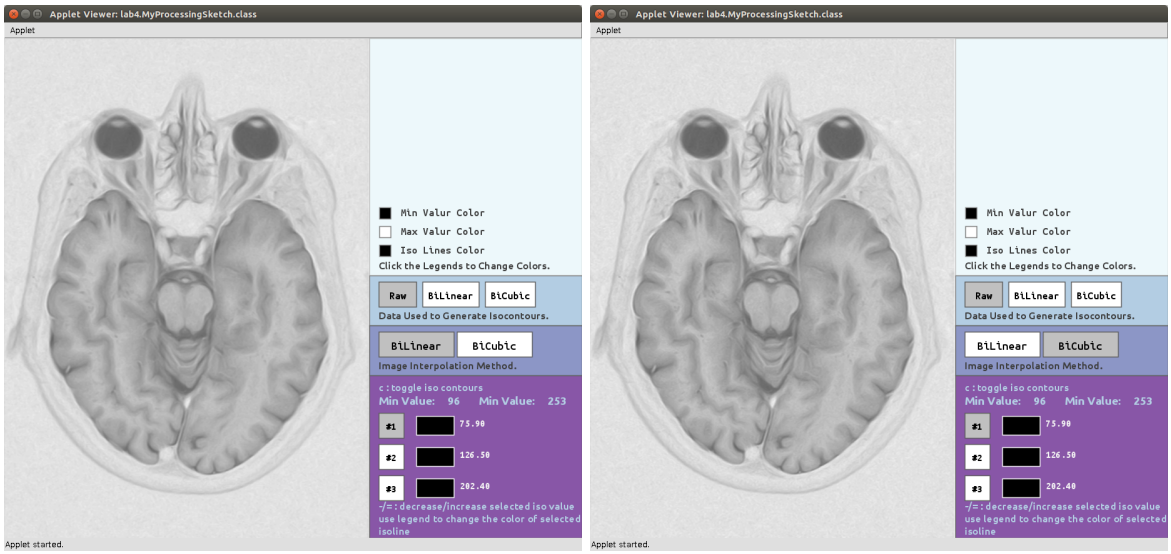
Interpolation (bilinear and bicubic)

Interpolation affects data dramatically if we enlarge it several times, in this case, the test data. One can still see the streak effects from the bilinear interpolation, partial reason is the trick of our visual perception, but also because of the interpolation itself. 8x8 crispy squares maintain their center value but lose the definition of edges heavily. In contrast,

bicubic interpolation takes further neighbors into account, it looks blurry, but results in better original appearance and edge definition.



The brain data contains sharp transitions, bilinear interpolation blurs the edges. Again, the cubic maintains edges much better by looking influence from further pixels.



Notes:

- When padding for bilinear interpolation, one only needs one extra data along each dimension, but for bicubic, one needs three.
- By taking influence from further pixels, bicubic interpolation actually denoises the image, e.g, the black area in the brain set looks smoother.

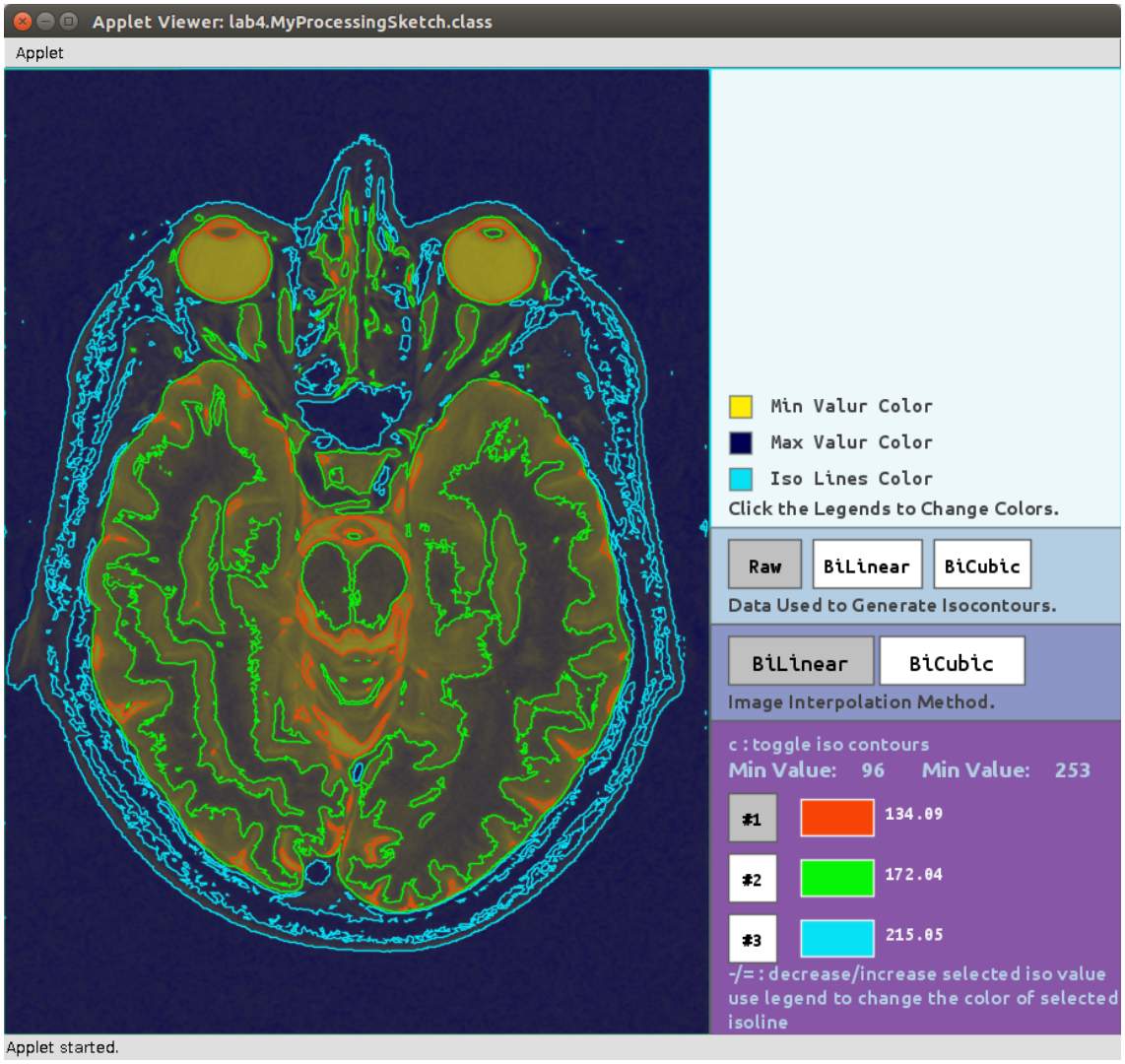
Isocontouring (on raw data and bilinear & bicubic interpolated data)

My implementation of marching squares consists of two parts, a class for marching and a structure for grid. A grid contains information such as the case and neighbors , so that the marcher can determine the contour easily by looking up the standardized data.

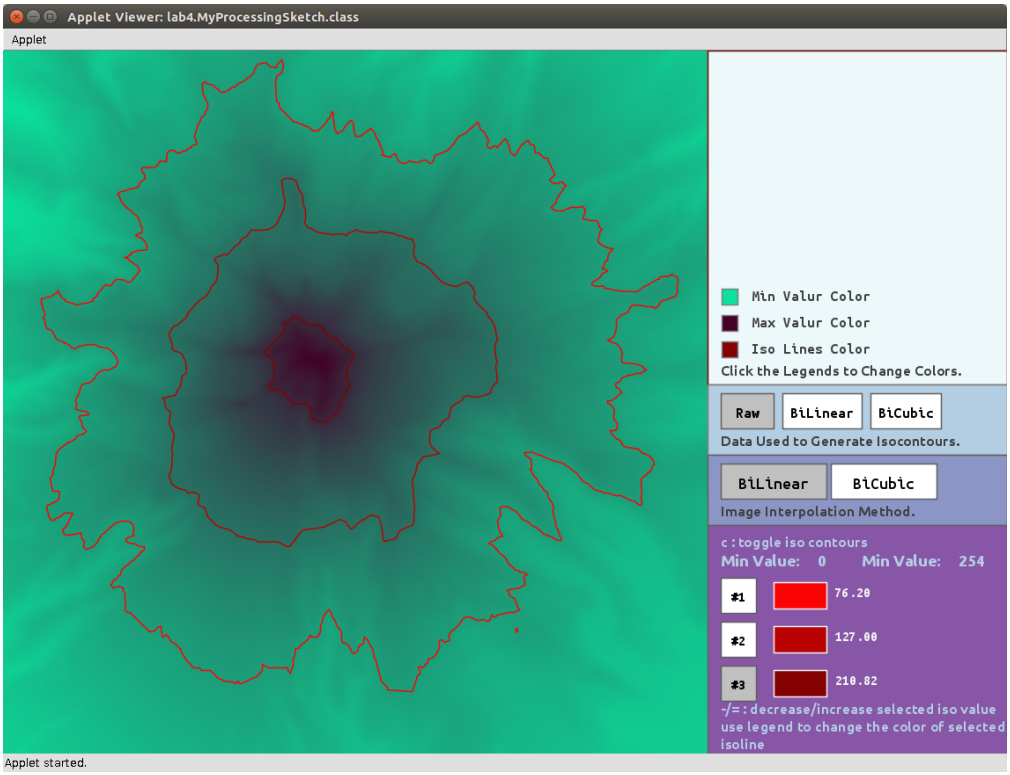
When looping through each cell in the grid, I start from this cell and draw lines through out cells that have connections with it and stop when it come back itself. This process can be done as many times as the number of iso value given, but getting slack if it goes over three on my laptop. So I pre added three iso lines here.One thing needs to mention is, by my implementation. The contour lines could go cross each other on the raw data of the test data, but it won't happen for any finer data.

Iso contour helps to find and denote the interested area (values), it's essentially an extremely fine color mapping. It also helps to visualize edges where color mapping can't do much without decent data values. From my opinoin, it's hard to say which one is better. Color mapping helps to overview the data and initiate a direction, iso contour helps to go further.

For the brain data, three values are interesting, 134 - just includes the eyeballs, from here I guess lower value means less dense; 172 - a threshold for the brain; 215 - a threshold for the whole head structure.



The really interesting data in the mtHood is 210, where it's a hole or a tip.



I also did iso contours on bilinear and bicubic interpolated data, both appear spikier than doing directly on the raw data. This is because the interpolated data is more localized.

Notes:

- I determine the two ambiguous cases by looking the cell's center value (average of it's four corners). For example, if it's case 5 and the center value is greater than the given iso value, then I flip it.

Extra

- Higher order interpolation - done.
- Isocontouring Interpolated data - done.
- Efficient coding
    - I cached data as most as possible
    - I don't update data if no action detected
    - I encapsulated most common functionalities into class, so the main applet looks much cleaner than before, although some hard code part for the UI design

Notes:

- It's interesting if we map the color using diverging color mapping and use the isovalue as the mid point
- I was gonna add mapping based on the range of input value, but I may break the rule.