Testing Protocol for Cone-beam x-ray Simulator

1. Purpose

Although the simulator has been constructed, various aspects of it must be validated and tested in order to make sure it is functioning as intended. The extended details of these tests are written below.

2. Test Variables and Reasoning

The following list contains aspects of the simulator that should be tested, along with detailed reasoning behind the tests:

- Test STL file read in functionality:
    - One of the main advantages of this x-ray simulator implementation is that it is able to read STL files and load them into MATLAB. It is important that the code implements the STL read in correctly and that any 3D object mesh is a proper representation of the object. If not, the 2D projections will be incorrect.
- Test voxelization of the 3D mesh:
    - Another important functionality of the code is that it is able to convert a list of vertices coordinates (representing the 3D mesh) into an actual 3 dimensional image array. Improper voxelization of the object will result in the 2D projections to contain errors.
- Test effect of distance on the size of the 2D projection:
    - Due to perspective projection geometry, objects that are closer to the x-ray source appear larger on the detector (similarly objects farther away appear smaller). This should be examined to confirm proper implementation of the cone-beam x-rays.
- Test for perspective distortions in the 2D projections:
    - In a proper cone-beam implementation, there should be image distortions in the 2D projection. These are due to the same reasoning in the test above: objects closer to the source appear larger while objects farther away appear smaller. Mainly, as the algorithm computes the projection of each image slice in the z-direction (away from source), they will reduce in projection size.

- Test for correct 2D projection direction:
    - The simulation assumptions and geometry are all based upon the fact that the code should be projecting in the z-direction (the detector is in the XY plane). If the code was not projecting in the correct direction, the final 2D projection image would be incorrect.

3. Test Methods

This section lists the methods of testing that should be utilized to validate the respective earlier simulation variables:

- Test STL file read in functionality:
    - Obtain several STL files for testing (sphere_ascii.stl etc.).
    - Run *READ_stl.m* to read in the STL files.
    - Use the MATLAB function *patch.m* to create a visualization of the 3D mesh object.
    - Confirm results with STL file sources.
- Test voxelization of the 3D mesh:
    - Open sphere_ascii.stl.
    - Run *VOXELISE.m* on the STL file with a 3D grid of at least 100x100x100 voxels.
    - Loop through the image slices of the object in any direction. Due to the object being a sphere, there should initially be a small circle that grows to a max circle size (centre slice of sphere) and then back to another small circle.
    - Since the circle shape in each slice is a binary image, we can use the MATLAB function *bwarea.m* to compute the area of each circle in each slice. Store these values.
    - Do this for each direction of the 3D image array, and compare the area values. They should be nearly the same for each direction.
- Test effect of distance on the size of the 2D projection:
    - Run *XraySim.m* on "sphere_ascii.stl" with a source-object distance value of 500.
    - Repeat this multiple times, varying the distance value each time (increasing or decreasing). Due to the object being a sphere, the projection should just be a circle each time.
- Test for perspective distortions in the 2D projections:
    - Create a new 3D image, with the following MATLAB code:
        - Object = zeros(100,100,100);
        - Object(20:80,20:80,20:80) = 1;

    - This object should be a cube in the center of the image array.
    - Run the x-ray projection on this object to create the 2D projection.
- Test for correct 2D projection direction:
    - Load "femur_binary.stl" into MATLAB. This is an excellent 3D object to test for correct projection directions as it has unique features making the X,Y and Z-direction projections very distinguishable from each other.
    - Use *VOXELISE.m* to turn the object into a 3D image array, and compute the parallel image projections for each Cartesian direction. This will be used to compare against the cone-beam projection from the MATLAB function *projection2D.m*.
    - Compute the cone-beam projection of the femur object, and compare against the parallel projections to determine accuracy.

4. Results

This section explains the possible results from the previously described test methodologies, and how to fix potential issues with the simulator.

- Test STL file read in functionality:
    - Depending on the patch image computed from the results of *READ_stl.m*, we can easily determine whether or not the object is correctly read into MATLAB.
    - If there is a problem with the results:
        - First check the STL file and the images from the download source. Confirm that it is different from the image in MATLAB.
        - If so, repeat this with other STL files to be sure the problem lies within *READ_stl.m*.
        - If *READ_stl.m* contains the error, work through the source code to find and fix the error.
- Test voxelization of the 3D mesh:
    - The area results from the binary image slices of the sphere should be nearly the same for each direction.
    - If the results are different then there is an issue with the ray-casting implementation in *VOXELISE.m*.
    - Work through the source code and continue the same testing until the error is fixed.
- Test effect of distance on the size of the 2D projection:
    - If the simulation is implemented correctly the circle should shrink when the distance value is increased (further from source) and grow when the distance value is decreased (close to x-ray source).

    - If the opposite happens:
        - Check the projection ratio in the function *projection2D.m* and invert it. This may solve the issue.
        - If this doesn't fix the problem, make sure the projection parameters are set properly (no negative distance measurements).
    - If the circle does not change size:
        - Check the function *projection2D.m* and determine that the 2D interpolation is set up correctly (confirm mesh grids).
        - Check the projection ratio.
- Test for perspective distortions in the 2D projections:
    - If the 2D cone-beam projection is correct it should have something similar to a "ringing" artifact along the straight edges of the square projection (due to the projection geometry).
    - If not:
        - Check *projection2D.m* and make sure that projection ratio is correctly calculated and applied.
        - Confirm correct 2D interpolation in case the above fix fails.
- Test for correct 2D projection direction:
    - Compare the 2D cone-beam projection of the femur object with the parallel projection in the Z-direction. They should be nearly identical.
    - If not:
        - Compare the 2D projection to the other projection directions. If it matches with one of the other two directions, then the projection implementation in *projection2D.m* is coded in the wrong direction.
        - Examine *projection2D.m* source code and change the direction back to Z.
        - If it looks nothing like any of the other projections, still check *projection2D.m*, as a major error in the implementation of the perspective projection mathematics is causing this.
        - Refer to the pdf: *Perspective Texture Mapping Part 1: Foundations by Chris Hecker* to confirm the mathematical implementation, and fix the error.

5. Conclusions

Once all of these tests are successful, the cone-beam x-ray simulator should be considered robust and ready for usage in further experiments.