Volume Rendering Documentation

First, there is definitions of point3d struct with three float fields representing x, y, and z coordinates and ray struct that contains two pointers to Point3d objects, representing the endpoints of a ray.

Then some helper functions performing vector math on Point3d objects, including subtracting, calculating the dot product and cross product of two vectors, and normalizing a vector.

The main functions:
1. rotation_transpose: It computes the transpose of the rotation matrix that would align the vector from p2 to p3 with the positive z-axis.
It does this by first determining the normal n of the vector from p2 to p3, then determining the vector u that is orthogonal to both n and the vector from p2 to p3, and lastly determining the vector v that is orthogonal to both n and u. The rotation matrix that results is then placed in a matrix.

2. ray_construction: With an image pixel position and camera settings, this function generates a ray in 3D space.
Pixel location (i,j), focal length, picture boundaries (xmin, xmax, ymin, ymax), a camera transformation matrix Mcw, and a viewpoint in world coordinates are all input parameters (vrp).
The result is a Ray object with the origin point (ray->a) and a normalized direction vector (ray->b).
The function first uses the camera transformation matrix and focal length to map the pixel position to a 3D point in camera coordinates, and then uses the viewpoint to convert that point to world coordinates.
The difference between the 3D point and the perspective is used to construct the direction vector, which is then normalized.
The Ray object stores the resultant origin and direction.

3. tri-interpolation: This function performs tri-linear interpolation to calculate the voxel value at a given point in 3D space
input arguments:
- data: 3D array representing the volume dataset
- curP: pointer to a point in 3D space where we want to calculate the voxel value
Output:
- val: interpolated voxel value at the given point.

It calculates the indices of the eight neighboring voxels. Then calculate the fractional distance between the point and the eight neighboring voxels. Finally, perform trilinear interpolation.

4. compute_shading_volume: Computes shading for a given light ray position and a computed CT volume
Parameters:
lrp[3]: an array representing the position of the light ray
ct[SLCS][ROWS][COLS]: a 3D array representing the computed CT volume
ip: intensity of the light source
shading[SLCS][ROWS][COLS]: a 3D array representing the computed shading
Returns: void

This function computes shading for a given light ray position and a computed CT volume.
It loops through all voxels in the CT volume and computes the length of the normal vector at each voxel.

If the length of the normal vector is below a threshold value, the shading is set to 0.
Otherwise, the function computes the normalized normal vector and light ray vector at the voxel, computes the dot product between them,
scales the result by the intensity of the light source, and casts the result to an unsigned char. The resulting shading values are stored in a 3D array.

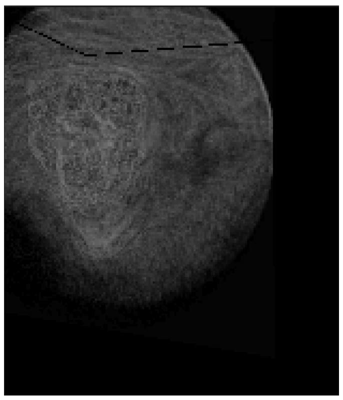
5. volume_ray_tracing: This function performs ray tracing through a CT volume to generate an image of the volume along the specified ray.
Inputs:
- ray: the ray to sample along
- ts: an array of two floats representing the start and end points of the ray
- ct: the CT volume to sample from
- shading: the shading information to apply to the volume
Returns:
- The generated image as an unsigned char value

6. ray_box_intersection: This function determines the intersection of a given ray with the bounding box of the volume. The bounding box is defined by the dimensions of the volume (ROWS, COLS, SLCS), where ROWS represents the number of rows, COLS represents the number of columns, and SLCS represents the number of slices.
The function takes in a pointer to a Ray struct, which contains the starting point of the ray (a) and its direction (b), and an array of floats ts[2], which will be populated with the intersection times of the ray with the bounding box.
The function returns the number of intersection points found, which will be either 0, 1, or 2.
The function first calculates the intersection times between the ray and each of the six bounding planes that define the box.
If an intersection point lies within the bounds of the volume (i.e. within the range of valid indices for each dimension), its intersection time is added to the ts array.
Finally, the function returns the number of intersection points found.

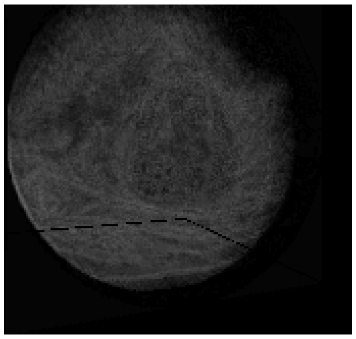
7. Main: The main function is using CT scan data to perform volume rendering. It reads in the CT scan data from a file named "smallHead.den", computes the shading volume for the data, and then renders an image of the data using volume ray tracing. The resulting image is saved to the file “outImage.raw".

Results:

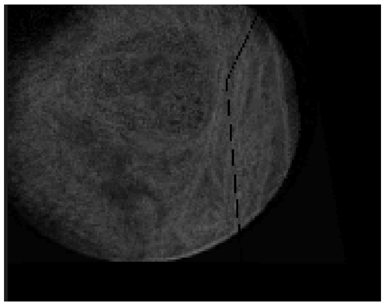
1. With default values in my model.h:



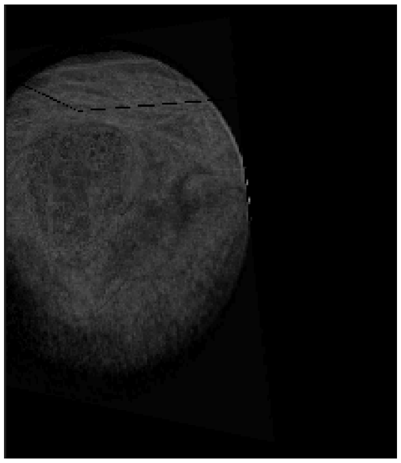
2. With VUP[1] = -1:



3. With VUP[2] = 1 and VUP[0] = VUP[1] = 0:



4. With VPN[1] = 64:



5. With VRP[1] = 0 and VPN[1] = 64: