Interactive Volume Rendering

Alicia Boucher
Department of Computer Science
Brown University

Abstract

We aimed to reduce the time and difficulty of a manual interaction task. This task involves aligning 3D point clouds that represent bones with CT data of hand bones in a functional position. Aligning the bones with the images is a time consuming and difficult task because the user must shift through a stack of hundreds of images and manipulate the point clouds by adjusting and entering numbers. The alignment needs to be accurate so that small changes of positions and rotations of bones may be observed within a subject over time. We aimed to provide a 3D comparison of the data in place of the 2D image stack and to add a viewport-based manipulation tool. Instead of aligning the point clouds with the 2D images, the users would align the point clouds with a volume rendering of the CT data. Users found the 3D manipulation useful and believed that the volume rendering helped them locate where the point mesh's orientation needed to be adjusted. They believed that the 2D images were still useful in aligning the point clouds because the clear 2D contours give clues on how the edges of bones need to be oriented.

1. Background

Osteoarthritis in the thumb carpometacarpal joint causes great impairment and pain to the hand. The disease is significantly associated with symptoms of morning stiffness, joint swelling, and nocturnal joint pain in women, but not men [1]. Researchers want to analyze the possible biomechanical factors of thumb carpometacarpal osteoarthritis in order to better treat the symptoms and disease.

In vivo longitudinal data is collected from patients with early stage osteoarthritis in the joint. The patients' wrists are scanned in a neutral position and during several functional tasks—such as grasping. A researcher will segment the neutral position data in order to generate point clouds of each bone in the neutral position. These bones and CT data from the same patient's wrist during a functional task are fed to an automatic registration program that determines how the bones are rotated and translated from the neutral position to the functional position. Afterwards, the researchers can gather data on the joint biomechanics and joint narrowing to better understand how the joint is affected by osteoarthritis. This understanding will be useful in designing better procedures for restoring the joint and for designing better implants than tested ones, which have had a high failure rate [2].

Before the data received from the automatic program can be analyzed for these purposes, it must be manually checked against the 2D CT data of the functional position. If some bones are misplaced, they must be corrected by a researcher using a program called WristViz. Correcting a single position can take up to five hours of human interaction with the program, so we aim to reduce time spent on this task.

2. Point Cloud Interaction

In the WristViz program, bones of the hand and wrist are represented as point clouds. The user can modify the orientation of each bone within the viewport space through spin boxes that list the translation and rotation angle along each of three axes. A 2D image stack of CT data can be shifted through to see the contours of the bones at various levels. The user edits the values in the spin boxes to align the bone with the contours.

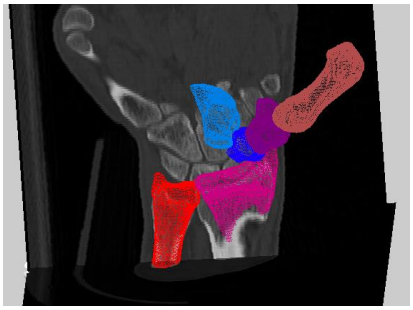

Figure 1: Colored point clouds against slice of CT data

Although spin boxes feature precise modification of data, shifting through numbers is not ideal for manipulating 3D objects because the user can only control rotation or translation along one axis at a time. Another disadvantage is that shifting through numbers in spin boxes cannot be done at an easily controllable rate. We believed a viewport based manipulator would make the task easier because it allows the user to rotate or translate along several axes at once if desired. 'Instead of having a speed limit on the changing of the numbers, the users can drag a manipulator quickly or slowly for more precision.

Problems arose in handling orientations of bones through a viewport based manipulator because the orientation is represented by three euler angles. This fact makes the data prone to gimbal lock, which should be avoided before it is passed to the registration program again. To handle this, the rotations of the manipulator are capped in regions that approach gimbal lock. This limits the possible orientations of the bone. The limitation is acceptable because local rotations of over 90 degrees in yaw are unlikely in the bones being studied.

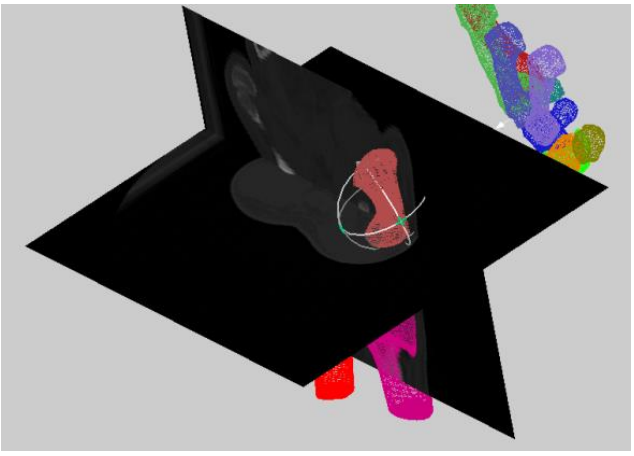

Figure 2: Coin3D manipulator around a bone

3. Volume Rendering

While the user adjusts the orientation and location of a bone point clouds, he must be shifting through the collection of 2D data to get a complete sense of how to orient the mesh. We believed that a 3D representation of the data would offer a more complete view of the data and would be easier to compare with point meshes.

We considered a volume rendering to be an ideal 3D representation for this task. With a volume, the user can adjust transparency, level of detail, rendering styles, and a transfer function at runtime. Control of transparency was thought to be important because a user may want to see the point cloud against an opaque object or be able to see all points aligning through a transparent surface. Control of level of detail is important in balancing the amount of detail that will yield accurate results and decent interactive performance.

The volume rendering is programmed with a texture based method. Several hundred proxy planes lying within a cube are textured with parts of a 3D texture. The planes are constantly shifting so they face the camera and together appear to be a continuous 3D object [3]. The program runs with OpenGL and simply takes advantage of OpenGL render states to ensure that the transparency of the volume is correctly rendered.

The CT data contains information about bone, fat, muscle, and other soft tissues. Only the bone data is relevant to the task. Thus, the transfer function on the texture is set to display only data that falls within 0.35 and 1.00 intensity. Although the default values provide a solid picture of the bones, it is possible for the user to adjust these thresholds at runtime.

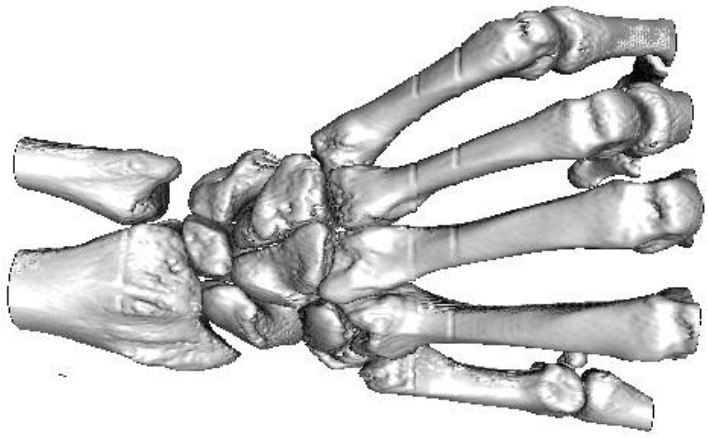

Figure 3: Volume rendering

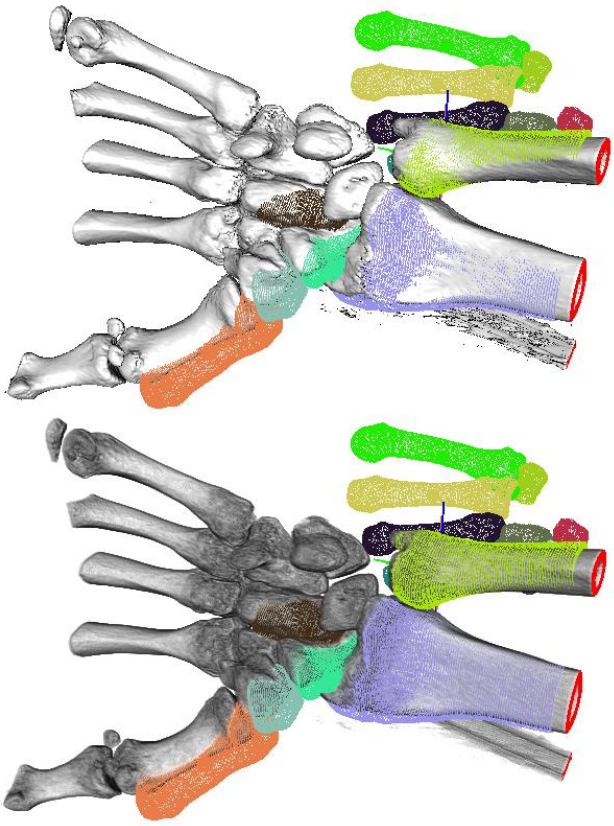

Figure 4: Mesh against opaque and translucent volumes

The volume rendering program has options to display the data as opaque or translucent. We believed that the opaque volume was easier to compare to the point meshes because it was easier to see one surface than many levels of surfaces. However, points on the mesh disappear into the surface of the volume when a bone is closely aligned with it. Thus having levels of translucency available could be useful in seeing how far within the volume the points are located.

3.2 Interactive Performance Rate

The program utilized the OpenGL programmable pipeline by performing rendering computations in a fragment shader. The shader handles the transfer function and lighting. Originally the shader had to perform seven texture accesses, one for the data at the location and six more to compute central gradients in each direction. With seven texture accesses multiplied by the number of planes and each pixel that contained a plane, the program ran at poor rates.

Number of Proxy Planes	FPS for Normals computed at run time	FPS for Precomputed normals
50	59.50	59.50
100	31.20	32.70
600	5.46	5.72
1200	2.70	2.80

Figure 5: Frames Per Second (FPS) comparison

Because the gradient data is static, it is possible to precompute the normals at load time and to store the information in a signed normalized texture. The shader can access a normal from this texture, reducing the total number of texture accesses from seven to two. The normals can be precomputed using a finite difference method with two central differences to more accurately estimate the gradients.

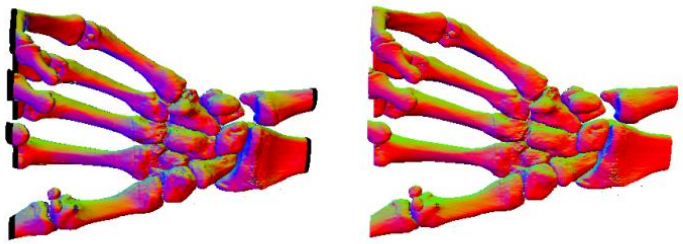

Figure 6: Precomputed normals and runtime normals

Another way that FPS was improved was by reducing the number of proxy planes drawn during user operations. This meant reducing the number of planes when the user is turning the camera or changing rendering parameters. When the proxy planes need no update, the screen can remain uncleared while displaying a high resolution volume, so that the program can still respond immediately to user input.

3.3 Slicing Data

While a volume shows the CT data in its entirety, it may introduce clutter during the alignment task. The researchers must adjust the placement of fifteen bone point clouds, and many of the bones in the middle of the wrist are very small and surrounded by several other bones. The important features of these bones might then be hidden. A feature to the program that allows the user to 'slice' the volume at specified

points along each coordinate axis can reduce the portions of the volume shown, removing obstructions to a bone of interest.

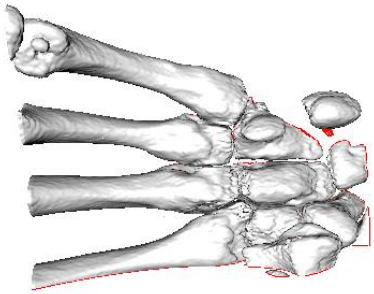

Figure 7: Reduced Volume

The researchers liked the clarity of the bone contours when viewing the 2D stacks of CT slices. However when a volume is sliced in 3D, the contour is unclear because it does not stand out from the pixels coming from inside the bone in neighboring regions. The contours can be highlighted, within the shader, along the sliced edges in an attempt to encompass the benefits of the 2D representation within the volume.

Figure 8: Highlighted contours

4. **Rendering Multiple Volumes**

In the case of representing same subject CT data in two different positions, the relative position of the wrist in each volume can be compared. By drawing the two different volumes to the same planes, a fragment will have access to both volumes and can have the ability to render fragments of intersecting bones differently through the fragment shader. In an opaque mode, the intersecting region can be seen as an outline along intersections and when the volume is low opacity the intersecting region can be seen through the bones. The shading normals from either source can allow the user to view the intersection as in association with either source.

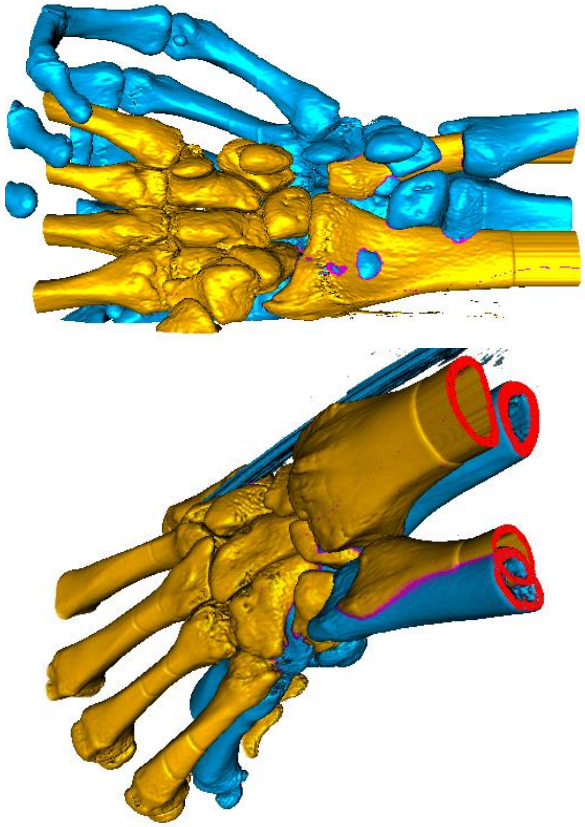

Figure 9: Intersecting regions highlighted in magenta

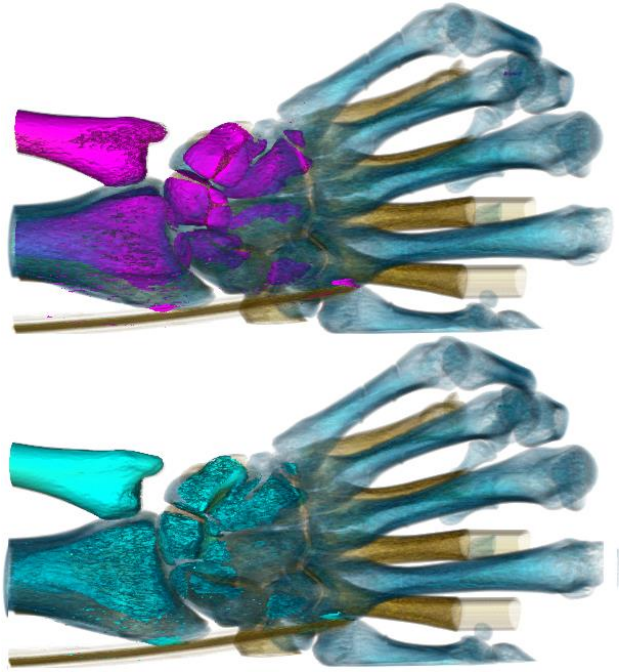

Figure 10: Intersections with normals from different sources

This addition is simple when considering data that exists in similar places within each volume. When

the volumes must be translated or rotated separately from the other, for a clearer comparison, operations on the mapping of texture coordinates to the planes must be computed first. The transformation can be passed into the shader as translation values and a representation of orientation. The transformation can be applied to the texture coordinates of a volume within the fragment shader.

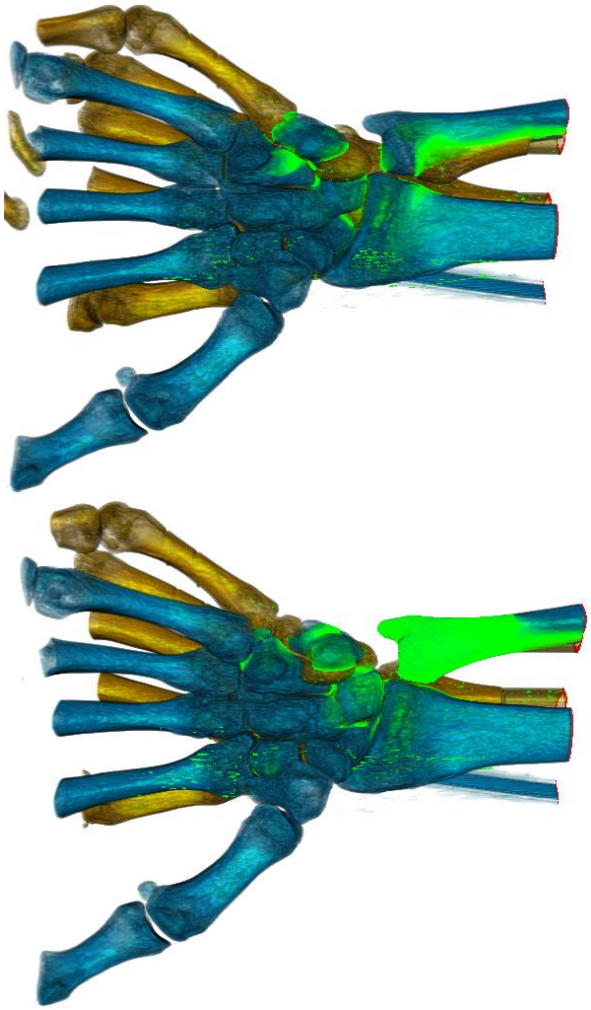

Figure 11: Alignment of the ulna by repositioning

5. Conclusions

The researchers found the 3D viewport manipulator useful. They viewed the volume rendering with the point cloud bones and determined that the volume rendering would be beneficial to the task. They agreed that the main use of the 3D representation would make it easier to initially identify on what bones the registration failed, and thus which bones need user correction. The researchers felt that 2D CT images were more useful than a 3D image when aligning the point clouds because the contours in the images are clearer. The 3D representation offers another means of comparing the point clouds to the data, and can be included in the process as an option.

6. Acknowledgements

I would like to thank Professor Laidlaw for his insightful advice in developing these volume rendering features and Trey Crisco for explaining joint narrowing and medical imaging projects.

7. References

[1] Acheson RM, Chan YK, Clemett AR. New Haven survey of joint diseases. XII. Distribution and symptoms of osteoarthrosis in the hands with reference to handedness.

[2] Hansen TB, Vainorius D. High loosening rate of the Moje Acamo prosthesis for treating osteoarthritis of the trapeziometacarpal joint.

[3] Ikits, M., J. Kniss, A. Lefohn, and C. Hansen. Volume Rendering Techniques http://http.developer.nvidia.com/GPUGems/gpugems_ch39.html