Collaborative view-aligned annotations in web-based 3D medical data visualization

Primož Lavrič, Ciril Bohak, Matija Marolt

University of Ljubljana, Faculty of Computer and Information Science, Ljubljana, Slovenia
pl9506@student.uni-lj.si, ciril.bohak@fri.uni-lj.si, matija.marolt@fri.uni-lj.si

Abstract - the paper presents our web-based 3D medical data visualization framework with emphasis on user collaboration. The framework supports visualization of volumetric data and 3D meshes in web browsers. The paper focuses on integration of user-shareable 3D view-aligned hand drawn or written annotations into the visualization framework. Annotations are created on separate transparent canvases which are aligned with selected views. View parameters are part of annotations and can be shared with other users over the network. Our implementation allows for real-time sharing of annotations during creation. Annotations from the same or different users can be overlaid within the same view. Annotations were implemented through adaptation of the framework's rendering pipeline, which allows for combining multiple visualization layers into a unified final render. View aligned annotations were added in addition to text annotations pinned to 3D locations on the displayed model. In the framework, users can list through all annotations, whereby upon selection of a 3D view-aligned annotation the camera is positioned according to the stored parameters and the annotation is displayed.

I. Introduction

Visualization of 3D data is an already well established way of supporting work in many different fields, including medicine. In the paper we are focusing on visualization of volumetric data, which can be obtained with techniques such as: Computed Tomography - CT [1, 2], Magnetic Resonance Imaging - MRI [3], Ultrasound [4] and Positron Emission Tomography - PET [5]. Different techniques are suitable for capturing details of different tissues. The common property of all volumetric data is that the data is presented as three dimensional scalar or vector field containing property values for individual blocks of the scanned volume.

Such data can be visualized with indirect or direct rendering techniques. In first case the data is first converted to 3D mesh models [6, 7, 8] and then rendered [9], while with direct rendering, different volumetric rendering techniques can be used [10, 11, 12].

Most of 3D medical visualization systems were developed as standalone applications and require high-performance hardware for real-time display of data. In the past, we have developed a web-based volumetric medical visualization framework - Med3D [13], which allows users to visualize volumetric data in a web browser. The framework exploits the use of local and remote processing power for processing as well as rendering purposes. It's user interface is presented in Figure 1.

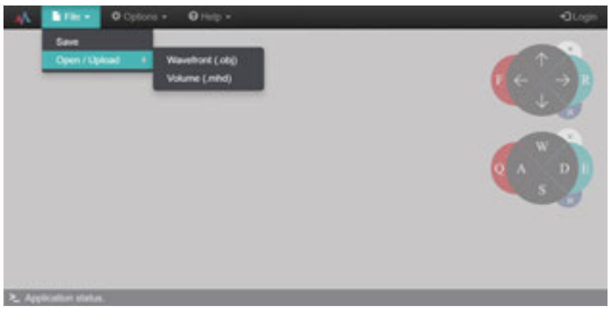

Figure 1: *User interface of Med3D - a web based volumetric data visualization framework.*

It is commonly accepted that user collaboration aids problem solving, and this is also the case in medical diagnosis, where second or even third opinions from experienced colleagues may be needed. In general, doctors with similar expertise rarely work in the same institution or even country, making collaboration slow and ineffective.

We have previously presented the benefits of remote collaboration integrated in our web-based 3D medical visualization framework [14] that allows for sharing: (1) visualization data, (2) camera view, (3) 3D localized user annotations and (4) text-based chat.

In this paper we present an extension of the Med3D framework. We first describe the implementation of render passes and render queue in Section II., in Section III. view-aligned hand-drawn annotations, in Section IV. the implementation of annotation sharing, in Section V. we describe how multiple users can concurrently produce annotations aligned with same view and in Section VI. we present the conclusions and future work.

II. Render passes and render queue

To be able to easily extend the visualisation framework and achieve good performance, it is of crucial importance that the underlying rendering pipeline is well designed and implemented efficiently.

In Med3D framework we extended the basic rendering with multiple render pass design. This design emphasises the deferred rendering approach and allows us to easily define the input data (uniforms, buffers and textures), the output (textures, screen) and the shader for each render pass. Data binding and shader selection is performed in the prepossessing function that executes prior to the rendering. This step also allows us to reflect the input data on the user input, the output data of previous render passes and the global state of the framework. Render passes can be grouped together in the render queue as shown in Figure 2.

Formed render queues allow us to execute the render passes in the desired order. Each render pass also has access to the queue's global intermediate render data which allows the data such as textures and other variables to be forwarded to the subsequent render passes. The last render pass can either output the texture to the screen or return it as a queue execution result along the global render data.

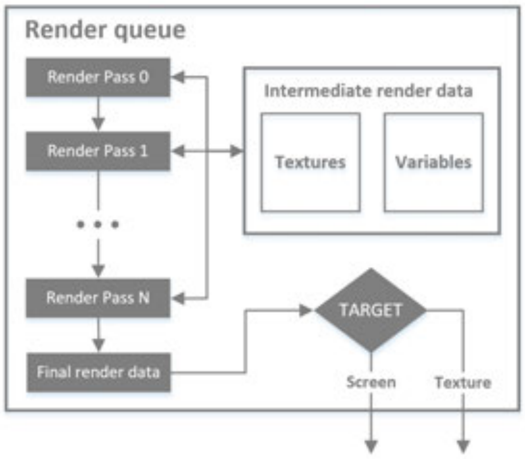

Figure 2: *Figure presents the structure of the rendering queue implemented in the Med3D framework.*

Our rendering pipeline design allows us to easily add new and expand the existing visualisation functionalities of the Med3D framework. Example of such functionality is the overlay multi-layer drawing which we integrated in the framework.

III. View-aligned hand-drawn user annotations

The Med3D framework allows users to add annotations on the displayed 3D data. Originally we implemented the textual annotations, which can be pinned on to desired spot on 3D data, making annotation connected with specific part of the visualized data. We have already presented such annotations implementation in [14] and an example of such annotations is presented in Figure 3.

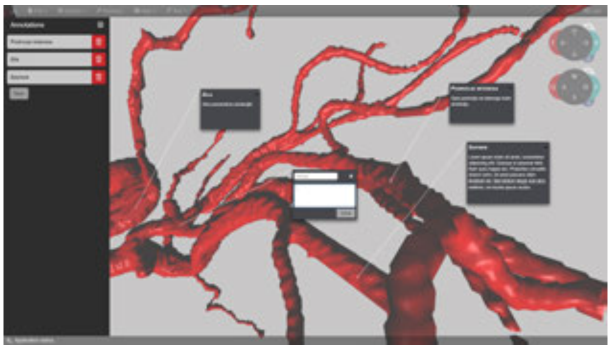

Figure 3: *Figure shows annotations pinned to the selected locations on the model of data.*

When we interviewed end users (doctors) about the initial implementation of annotations in the Med3D framework, they suggested that the implemented annotations are good, but that we should add the possibility of hand-drawn sketches on top of the visualized data.

We therefore expanded the Med3D framework with a drawing functionality where the user can either use a mouse, drawing tablet or touch screen to sketch the annotations. To start the sketching, the user first needs to create a new drawn annotation in the annotation sidebar shown left in the Figure 4. This sidebar contains a list of all the annotations that can be shown, as well as the brush tools such as color, thickness and hardness selector. To create a new drawn annotation user first needs to align the view to capture the point of interest and then create a new annotation. Upon creating the annotation the view is fixated on the current position and camera parameters (position and rotation) are stored so that the view can later be realigned. If the user wishes to view any of the previously created annotations, they can select them from the sidebar. Upon selecting the annotation the view is animated to the right orientation by interpolating the camera parameters (position, rotation) from the current to target values. After the camera is correctly positioned and oriented, the drawn annotation starts rendering on top of the data. This gives a smooth user experience when reviewing previously drawn annotations.

Each annotation can hold up to 20 drawing layers. Each layer holds a texture on which the data is rendered. When rendering the final render to the screen these textures are overlaid based on the order (bottom to top) in which layers are listed in the sidebar. The user can reorder this list using the arrows that appear next to the listed layer while hovering over it with the cursor, consequently changing the overlaying order. Layers can be added, deleted, renamed and hidden/shown using the provided user interface. To begin sketching, the user needs to select the target layer by pressing the ”pen button” present on all the layers that are not hidden.

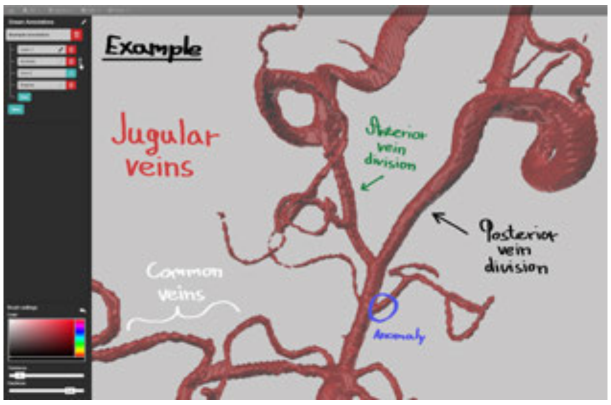

Figure 4: *Figure shows several hand-drawn annotations sketched on different layers and on the left the drawn annotations side bar. On the top of the sidebar is a list of annotations and a list of layers for the selected annotation. On the bottom are brush tools that are used to configure brush color, thickness and hardness.*

The user can draw by dragging the mouse or pen across the canvas. While dragging, a line segment is drawn between each pair of points representing the current and previous cursor position. To draw a line segment, we need to transform the points to the texture coordinate system and pass them to a shader which renders the line segment to a texture with the selected color, thickness (determines line width) and hardness (determines the distance from the line after which it starts to fade off). Color is selected from a color picker located in the bottom part of the annotation sidebar (brush settings) which also holds two sliders that are used to configure thickness (ranging from 1 to 32 pixels) and hardness (ranging from 0 to 1 where 0 represents the maximal fade off).

Because we might need to redraw the lines (in case of window resizing, undoing and annotation sharing) we store the points in a line structure. This line structure represents the combined line segments from cursor press to cursor release and their color, thickness and hardness. Each layer may contain multiple lines. We also need to normalise the stored points with the current aspect ratio as the aspect ratio might not be the same after resizing or on a device of a different user with whom we shared the annotation. We only need to normalise the x coordinate as the view projection is set so that it scales the height to fit the whole canvas whereas the width must always be equal to the height to represent all of the coordinates in a normalised space. Normalised x position can be obtained as $x_n = (x - 0.5) * w/h$ where x represents the position in a texture coordinate system and w and h represent the current width and height of the canvas. When we need to redraw the point, the normalised position x_n is transformed back to the texture position as $x = x_n * w'/h' + 0.5$ where h' and w' represent the new canvas dimensions. Using this process we can store the line segments that are invariant to the screen aspect ratio.

Any layer can be redrawn at any time using the stored information. We do the redrawing using multiple render passes where in each pass up to 251 line segments are drawn. This limitation comes due to the fact that we can only pass up to 1024 float uniforms into the shader to support all of the devices that are compatible with WebGL 2.0[1]. But even with this limitation the redrawing process is still very fast and the redrawing itself does not need to occur very often so it does not affect users on slower devices.

Our implementation of the drawn annotations allows users to easily manipulate their sketches. It is designed so that it is intuitive and easy for the user to add or delete the annotations or the layers. This results in good user experience and can be easily extended to allow for sharing of annotations between users.

IV. Annotation Sharing

The Med3D framework already has many user collaboration functionalities such as the sharing of visualization data, views and text annotations. Sharing of hand drawn annotations is implemented as an extension of the latter functionality. It allows users to present their opinion in an intuitive way and share it with others.

Similar to other Med3D collaboration functionalities, sharing of hand-drawn annotations is done over a remote server on which all of the shared data is stored for easy and fast access. To start sharing the annotations, users must first create or join an existing session. This is already supported by the Med3D framework. Creating a session allows multiple users to view and interact with the same data in real time.

When the user creates a new session, all of the already existing annotations are uploaded to the server. The data consists of a list of annotations where each annotation contains a title, camera parameters (position and rotation) and the list of all layers. Each layer also contains a title, a list of lines where each line consists of color, thickness, hardness and points. Because the drawn annotation data is composed of only Javascript objects and primitives, we can easily send the data to the server using Websockets[2]. Because of the simplicity we're using the Socket.io[3] framework, which handles the transmission of binary data in an efficient manner. After all the data are uploaded, other users may join the session. When a new user joins, the session data (visualization data and annotations) are downloaded from the server. The downloaded annotations and layers are stored and equipped with an additional field that holds the username of the owner. By default the shared annotations are not shown right after they are downloaded. The user first needs to select them for display. At this point the layers are rendered to a new texture that binds to each shared layer and is used for all of the subsequent drawing. Because the line points are aspect ratio normalised, we can easily transform the positions to match the window aspect ratio.

After the user is synchronised with a server, he may add new annotations or layers to his own or to the shared annotations. All the changes that are made are being recorded and sent to the server in a dynamic time interval based on connection quality. After the server receives the changes it applies them to its copy of the data and broadcasts them to all of other session users. This reduces the work of the clients as they only need to send the data to the server and allows for new users to get the data directly from the server without requesting them from the session host. The whole communication process is presented in the figure 5.

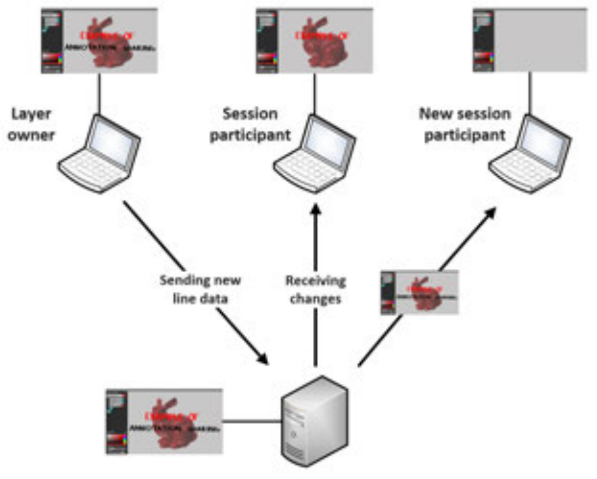

Figure 5: *Figure shows the communication process of sharing the hand-drawn annotations. On the top left there is the session host, which already synchronised the data with the server (bottom) and is now sending recent annotation changes. In the middle (top) there is a participant of the session that already downloaded all of the data and is now listening for changes as well as sending his own. On the top right is a new session participant that is downloading all of the latest data from the server without straining the session host.*

[1]https://www.khronos.org/registry/webgl/specs/latest/2.0/
[2]https://www.websocket.org/
[3]http://socket.io/

Our implementation of communication used for remote collaboration allows users to share their data and annotation changes in real time. This provides good user experience and efficient collaboration between multiple users with very little delay mostly dependent on the quality of the network connection.

V. Collaborative annotation

The most important aspect of hand-drawn annotations is that the users can make short notes and markings on the visualization, as well as interact with other users in real-time. This is also emphasised by allowing multiple users that are participating in the same collaboration session to overlay their layers and combine their sketches. Layers of current and all other users are listed together below the corresponding annotation. Each user can discern layers of other users by the user button that replaces the delete button. Hovering over this button displays the username of the layer owner in a tooltip as shown in image 6. Each user can, independently of other users, select the layers that he wants to see by clicking on the sidebar layer item. Users can also order the layers of each annotation using the arrow keys that appear while hovering over the layer to produce the desired overlay order. This allows for comparison of the data present on multiple annotation layers. Users can also use different colors, thickness and hardness allowing them to more easily emphasise the importance of different parts of the annotation as well as to distinguish between different parts of annotation.

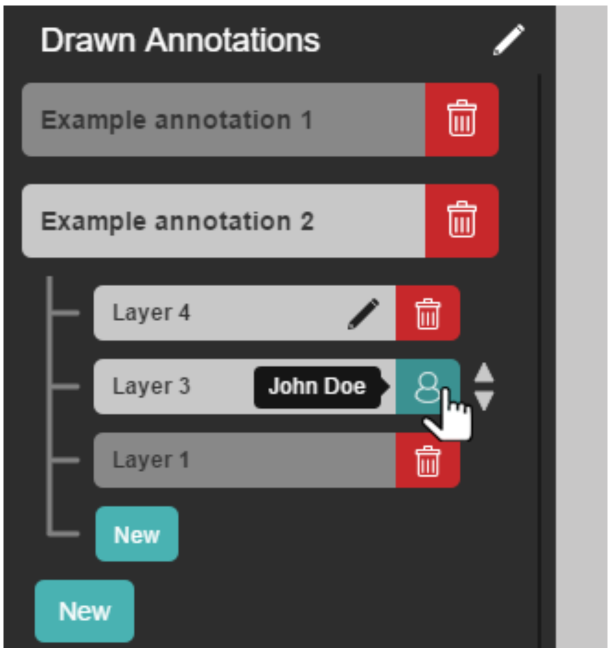

Figure 6: *Figure shows a list of drawn annotations in the sidebar. The second annotation in the list is selected and its three layers are displayed. Two layers are local and belong to the current user, while and one is shared by the user "John Doe".*

Because the layers are updated with short delay, users can see the changes being made in real-time. They can thus interact with each other while sketching with either concurrent drawing on different parts of the visualization or chatting via the text chat provided by the framework. In the future we also intend to add a voice chat that will in combination with hand-drawn annotations further improve the usefulness of the framework.

VI. Conclusion

In this work we presented an implementation of view-aligned hand drawn annotations into our Med3D visualization framework. The sharing of such annotations greatly improves the usability of remote collaboration and is an intuitive way for the end users (doctors) to express their views.

The next step in our work will be to evaluate the annotation interface with the doctors and implement improvements based on their feedback. We also aim to add a voice chat to further enhance the collaboration options.

References

[1] C. R. Crawford and K. F. King. Computed tomography scanning with simultaneous patient translation. *Medical Physics*, (17):967 – 982, 1990.

[2] W. A. Kalender, W. Seissler, E. Klotz, and P. Vock. Spiral volumetric CT with single-breath-hold technique, continuous transport and continuous scanner rotation. *Radiology*, (176):181 – 183, 1990.

[3] P. A. Rinck. *Magnetic Resonance in Medicine. The Basic Textbook of the European Magnetic Resonance Forum. 9th edition*, volume 9.1. TRTF, 2016. E-Version.

[4] D. Krakow, J. Williams, M. Poehl, D. L. Rimoin, and L. D. Platt. Use of three-dimensional ultrasound imaging in the diagnosis of prenatal-onset skeletal dysplasias. *Ultrasound in Obstetrics and Gynecology*, 21(5):467–472, 2003.

[5] J. M. Ollinger and J. A. Fessler. Positron-emission tomography. *IEEE Signal Processing Magazine*, 14(1):43–55, Jan 1997.

[6] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, August 1987.

[7] D. Lesage, E. D. Angelini, I. Bloch, and G. Funka-Lea. A review of 3d vessel lumen segmentation techniques: Models, features and extraction schemes. *Medical Image Analysis*, 13(6):819 – 845, 2009.

[8] M. T. Dehkordi, S. Sadri, and A. Doosthoseini. A review of coronary vessel segmentation algorithms. *J Med Signals Sens*, 1(1):49–54, 2011.

[9] W. J. Bouknight. A procedure for generation of three-dimensional half-toned computer graphics presentations. *Commun. ACM*, 13(9):527–536, September 1970.

[10] R. A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. *SIGGRAPH Comput. Graph.*, 22(4):65–74, June 1988.

[11] M. Levoy. Display of surfaces from volume data. *Computer Graphics and Applications, IEEE*, 8(3):29–37, 1988.

[12] E. P. Lafortune and Y. D. Willems. Bi-directional path tracing. In *Proceedings if Third International Conference on Computational Graphics and Visualization Techiques (COMPUGRAPHICS '93*, pages 145–153, 1993.

[13] C. Bohak, P. Lavrič, and M. Marolt. Web based visualisation framework with remote collaboration support. In *Proceedings of 25th ERK 2016*, pages 43–46, Portorož, Slovenia, September 2016.

[14] C. Bohak, P. Lavrič, and M. Marolt. Remote interaction in web-based medical visual application. In *Human-computer interaction in information society : proceedings of the 19th International Multiconference Information Society - IS 2016*, pages 5–8, Ljubljana, Slovenia, October 2016.