Low latency 3D image streaming in medicine

Miklós VINCZE
BioTech Research Center, EKIK
Óbuda University
Budapest, Hungary
miklos.vincze@uni-obuda.hu

Bence BIRICZ
BioTech Research Center, EKIK
Óbuda University
Budapest, Hungary
bence.biricz@stud.uni-obuda.hu

Abdallah BENHAMIDA
BioTech Research Center, EKIK
Óbuda University
Budapest, Hungary
benhamida.abdallah@nik.uni-obuda.hu

Miklós KOZLOVSZKY
John von Neumann Faculty of Informatics
Óbuda University
Budapest, Hungary
kozlovszky.miklos@nik.uni-obuda.hu

***Abstract*—3D visualization has been present in medicine for several decades, just think of the Computer Tomography (CT) imaging modality, which is capable of creating 3D images of the examined object by determining the radiation attenuation coefficient. In addition to CT equipment, there are many solutions capable of collecting and displaying various 3D data (Magnetic Resonance Imaging [MRI], Positron Emission Tomography [PET], Optical Computer Tomography [OCT]). In digital pathology, 3D visualization is not nearly as widespread as the previously mentioned solutions. One of the main reasons for this is the size of the digitized pathological samples and the hardware resources required for their 3D visualization. In the paper, we present a client-server architecture that can provide a solution to the existing hardware limitations during the 3D visualization of digitized pathological samples. In the paper, we present the basic structure of our solution, several main functionalities, and the testing of the server software we developed.**

Keywords— Visualization, 3D visualization, multi-user, digital pathology, network communication, video streaming in medicine, virtual user generation, hardware utilization testing

I. INTRODUCTION (*HEADING 1*)

Nowadays, visualization has become an indispensable function in medicine, without which neither pathologists nor other doctors can make a proper diagnosis. The software-supported evaluation of digitized medical samples, be it Computer Tomography (CT), Magnetic Resonance Imaging (MRI), or even digitized pathological tissue samples, has become part of everyday work. 3D visualization programs, which make digitized medical samples available for examination in a 3D environment after registration of the samples, must be treated separately from "simple" 2D visualization programs. In the case of certain imaging modalities, 3D visualization is already part of routine diagnostics, such as:

- Computer tomography
- Magnetic resonance imaging
- Optical coherence tomography

However, the 3D reconstruction [1-2] of digitized serial sections in a suitable way is still a research area in digital pathology [3]. Several factors make the 3D visualization of digitized pathological samples, so-called Whole Slide Image (WSI) samples, difficult. These include, among others:

- Large data size
- Coded data
- High resolution
- A Large number of samples per serial section

Due to these aggravating circumstances, if we want to develop 3D visualization software for displaying digitized pathological samples, we must expect that only the most powerful hardware components currently on the market will be needed.

Unfortunately, at the moment, hardware capable of processing hundreds of WSIs for 3D visualization in virtual reality is too expensive to be widespread, and it is not even expected that everyone has such powerful hardware [4].

One of the technologies offering a solution could be network streaming, with the help of which we would be able to transmit the data from a central computer to the users so that the users would not have to invest in expensive devices [5-8]. If the user wants to see the digitized medical samples processed by the server, he would be able to do so on his own, everyday computer.

A. 3D graphics engines

The use of 3D graphics engines is common in everyday work in certain areas, such as architecture or game development. The application of 3D graphics engines in areas such as medical informatics provides an opportunity to apply the experience accumulated in the field of game development and entertainment software development over the past 40 years in digital medicine.

Nowadays, several such 3D graphics engines are available on the market, which differs in the functionalities they implement, the supported programming languages, licensing, and the supported target platforms. To implement our software, our team chose the Godot graphics engine, as it provides many programming languages for development, such as C#, and C++. Another advantage in addition to the Godot graphics engine is that it supports multiple target platforms such as Windows, Linux, and Macintosh [9-10].

B. Video streaming

Nowadays, video streaming defines our everyday life, whether it is entertainment or work [11-13]. Real-time video streaming techniques also differ from non-real-time in their basic operation, but one of the most important differences is that real-time video streaming is much more sensitive to delay, jitter, and packet loss [14-17]. To provide the best possible user experience, developers use various adaptive streaming techniques, such as Dynamic Adaptive Streaming over HTTP (DASH) [18] or Active Queue Management (AQM) [19] [20]. Today, DASH-based systems are the basic solution for non-real-time and real-time video streaming systems and it is a common solution to use DASH systems in combination with AQM solutions. AQM currently has 4 common solution methodologies. These are Random Early Detection (RED) [21], Adaptive Random Early Detection (ARED) [22], Controlled Queue Delay (CoDel) [23], and Proportional Integral controller Enhanced (PIE) [24] [25].

II. Visual video streaming in medicine – State of art

A. *Video streaming in surgery*

One of the areas in which the use of real-time video streaming has become popular is to increase the efficiency of surgical interventions. The use of smart glasses in this area was obvious, as these devices provide surgeons with a convenient way to utilize the new information that they receive through real-time streaming techniques. The first application of smart glasses using a real-time streaming technique within the framework of surgery was published in 2013 in the following article [26], and it is still a popular topic today [27].

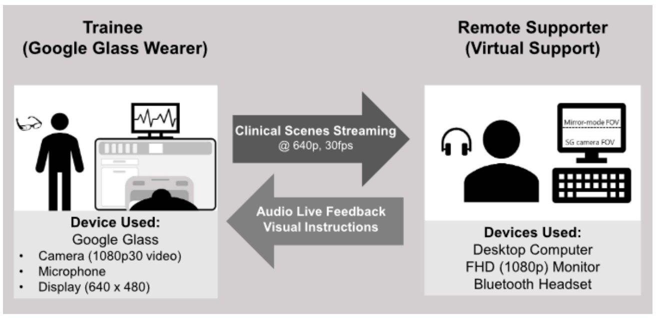

Fig. 1. An example usage of network streaming in surgery [27].

In Fig. 1 we can see, that combining smart glasses with network real-time streaming technology can provide advantages in surgery such as intraoperative counselling or innovative training of surgeons.

The application of the network streaming technique in surgery is not limited to different smart glasses or different Extended Reality (XR) solutions. Several types of research have been carried out in the past, in which the network video stream is monitored by an observer at another point of the earth with the help of a traditional smart device. The [28] publication presents such a solution, in which the authors also present the transmission of sound data, with the help of which the surgeon can receive sound-based feedback from a remote colleague based on the real-time image material.

The COVID-19 epidemic has shown that, among other things, the use of real-time network streaming techniques can play a major role in surgical education. The publication [29] shows an example of this.

B. *Video streaming in digital pathology*

The application of real-time streaming technology in digital pathology, including in the field of telepathology, results in a solution that provides pathologists with the opportunity to remotely diagnose high-resolution digitized pathological samples. An example of such a solution is the Remote Medical Technologies™ software, which can be used to perform an online consultation using real-time video streaming [30-31].

In the solution, 2 pathologists have the opportunity to establish a joint diagnosis, moreover, in such a way that the large digitized samples are only available to one of them. In contrast, the other user can see the samples displayed by the first user in the form of a real-time video stream.

Of course, COVID-19 also showed in pathologist education that the use of network data streaming technologies is essential in the future. The publication [32] shows an example of this. In the solution described by the authors, the user can start a real-time stream for several clients, thereby sharing the pathological samples with other users and sharing different cases with the students.

III. Our implementation of visual data streaming using virtual reality

The basis of our implementation is that, unlike other streaming techniques, we do not transmit the entire 3D environment to the user, but always only the 2D image that would be in front of his eyes. With this technique, we can reduce the amount of data to be transmitted over the network, which leads to the fact that we can provide users with a more continuous, higher-resolution video stream. The basis of the technique implemented by us is presented in Fig. 2. The meaning of the abbreviations in Fig. 2 are:

- B_n: The buffer at the client side, where the independent clients store their streaming data, which came from the server. The size of the buffer for the different users can be different.
- Vf_n: The visualization frequency of the different users. This is the speed, whit which the user visualizes the data, that the server sent. The visualization frequency depends on the network quality of the user. This variable can have different values for different users.
- Af_n: The asking frequency of a given user. This variable describes how fast the user asks for new data from the server. If the network quality of the client is good enough he can ask with a higher frequency, which will lead to a smoother stream on the client side. This variable can have different values for different users.

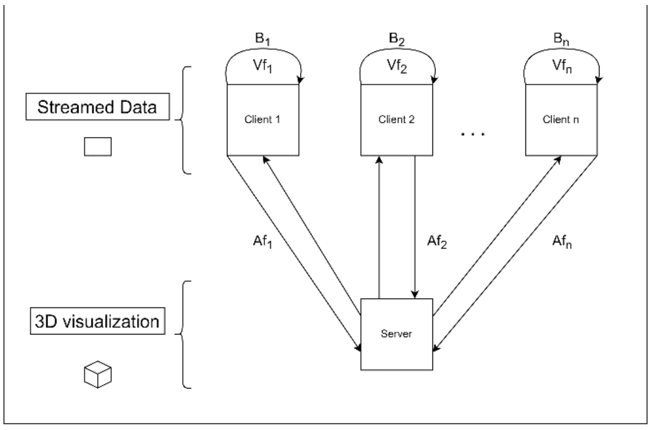

Fig. 2. The basic demonstration of our solution.

As shown in Fig. 2 in our solution only the server, i.e. entity capable of processing and displaying 3D data. To optimize the network data traffic, the individual clients receive from the server only the 2D image they currently need of the 3D world existing on the server. With this solution, users will always see that they are in an actual 3D world, and they will see the 3D digitized medical sample displayed on the server, but all resource-intensive calculations will run on the server.

A. The implementation of the client, and server architecture

To be able to implement the described solution, we had to plan the architecture of the server and the client, because in our solution these two components are developed as separate executables.

1) The implementation of the client

The client program can be run on any computer that can connect to the network from which the server is also accessible. We had to implement several basic functionalities on the client for our program to run stably. These were:

- Connection of the client to the server
- Client-side buffering rules
- Determination of delay compared to the server
- Transmission of movement data from the client to the server

Each user is represented to other users as a so-called avatar so that they can distinguish from each other.

It should be mentioned that even though the real 3D rendering is done on the server, the movements performed by the user are transmitted to the server, and then the user receives a new image from the server corresponding to his movements. This implementation allows the user to fully control his local movement, but the calculation required for 3D visualization remains on the server side.

2) The implementation of the server

For the implementation, we used the previously presented graphics engine, to exploit the possibilities that a graphics engine can offer regarding 3D visualization. We also made great use of the so-called camera class provided by the Godot graphics engine when implementing the server.

Every time a new user connects to the server, we create an instance of the mentioned camera class. These pediments indicate the position of the users in the central 3D environment to which all users connect. When one of the connected clients requests a new image from the server, the server receives the image from the camera belonging to that client. This solution enables all calculations related to the 3D environment and digitized medical samples to run on the central server, but all connected users can get a real-time view of it.

B. The implementation of the buffering rules

For our system to function stably even if certain packets do not arrive on time or not at all, we used a so-called ring buffer technique, which provides the opportunity for the clients to store the frames sent from the server to the clients, so the clients can provide the user with a stable service despite fluctuations in the quality of the internet.

The difference between the currently displayed frame index from the buffer and the latest frame index received from the server must always fall within a specific range. If this is not the case, one of the 2 buffering rules implemented on the client must be applied, as follows:

- If the difference between the indexes decreases, the client must slow down the display of frames received from the server. This results in the distance between the indices returning to the normal range.
- If the difference between the indexes becomes too large (larger than the 90% of the size of the buffer), the client must slow down the frame request from the server. This will cause the distance between the indices to decrease and then return to the normal range.

By applying the current buffering rules, our software can operate stably even with non-constant network parameters and can maintain the client-server architecture, in which the server provides a continuous video stream to the connected clients.

C. The implementation of the resolution changing in runtime

The current solution of our software includes functionality with the help of which, if the quality of the connection between a client and the server begins to deteriorate, the client will continue to receive frames, but with a lower resolution. This solution makes it possible for minor fluctuations in the network not to hinder the smooth use of the software.

The basic idea of the function is to examine the delay between the server and the client at given intervals. If this is greater than a certain threshold value, the resolution of the frames provided by the server to the client must be reduced. Otherwise, if the delay is smaller than a certain threshold value, the resolution of the frames received by the client from the server can be increased. The current version of our implementation supports 3 predefined resolutions a low, a normal and a high resolution. The different video stream resolutions can be seen in Fig. 3.

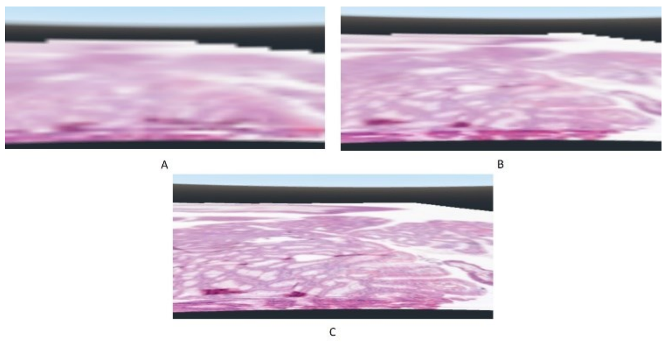

Fig. 3. The different video stream resolutions on the client side. A: Low-resolution video stream in case of high latency between the client and the server. B: Medium-resolution video stream in case of medium-quality latency between the client and the server. C: High-resolution video stream on the client side in case of low latency between the server and the client.

D. Implementing the virtual client generation

This research is an extension of the authors' previous work [33]. In order to keep the minimum computational resources as low as possible, the client receives the data in the form of a video stream from the 3D world created on the server. In order to provide the client with the appropriate video stream, the server must perform several steps when the client connects. These are:

1. Create a camera on the client server.
2. Set the resolution of the user's camera.

3. Create a client's avatar.
4. Placing the client in the 3D space created on the server.
5. Translation of the generated virtual users to the 3D model.

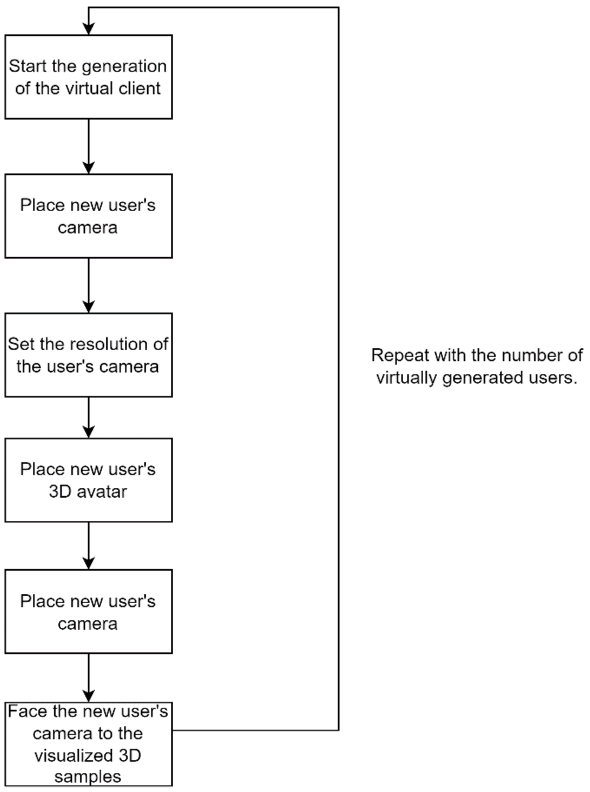
Start the generation of the virtual client
Place new user's camera
Set the resolution of the user's camera
Place new user's 3D avatar
Place new user's camera
Face the new user's camera to the visualized 3D samples
Repeat with the number of virtually generated users.

Fig. 4. The steps to generate virtual users in the server program.

By repeating these steps, a virtual system can be generated in which any number of virtual users can be created. This can be seen in Fig. 4. Virtual users differ from the actual users connected to the server in that they do not increase network traffic. This may be because the image data associated with virtual users is not sent out by the server to any recipient. This allows us to test the hardware usage of the server with different numbers of users, without the need to transmit network packets or without actually having hundreds of devices connected to the server.

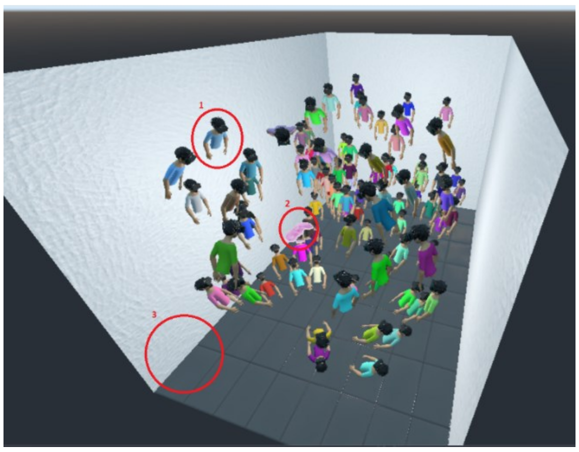

Fig. 5. Our server software with 100 virtually generated users.

There are several important details to highlight in Fig. 5. The first is that all virtual users are generated so that their corresponding camera is always facing the 3D medical sample. This is necessary because it is closer to a realistic usage and because what the user is looking at affects the server hardware utilization. Several details are numbered in the figure. These are:

1. 1 out of 100 virtually generated users.
2. A medical sample visualized in 3D.
3. The virtual examination room, where users can make a diagnosis together.

IV. Experimental Validation of Our Solution

A. *Testing the resolution changing*

To test the runtime, and change of the stream resolution, we set such threshold values for the change of resolutions during runtime, with which we could simulate all 3 implemented streaming resolutions. The results of the testing can be seen in TABLE I. It can be seen from the table that when the delay is greater than 100 milliseconds, the server starts providing a lower-resolution stream to the client in question, on the other hand, when the delay falls below 15 milliseconds, the client will receive a higher resolution video stream get from the server. Between the two values, the server provides the video stream to the client at an intermediate resolution.

TABLE I. The results of the resolution testing based on the latency change.

Index	Latency	Resolution
0	20	1
1	14	2
2	12	2
3	140	0
4	632	0
5	10	2
6	577	0
7	12	2
8	22	1
9	12	2

In TABLE I, in the resolution column, 0 indicates a low-resolution stream, 1 indicates a medium-resolution stream, and 2 indicates a high-resolution stream.

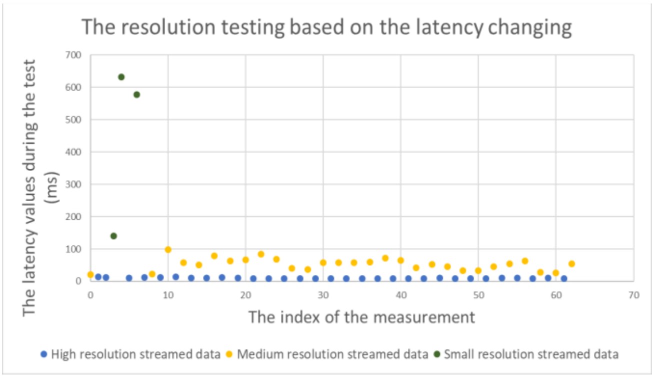

Fig. 6. The resolution changing based on latency fluctuation.

In Fig. 6, the different colours represent the streamed data of different resolutions depending on the delay. As shown in the figure, if the delay is high and the streamed resolution is low, these cases are shown in the image with black circles.

B. Testing the buffering rules

To be able to test the correct functioning of the buffering rules in a suitable testing environment, we have created methods with which we can manually speed up the video stream display speed of individual users, as well as the request frequency of individual users, which request new frames to be displayed to the server. By changing these two attributes at runtime, we were able to move our program from the stable state during testing, so we were able to trigger the activation of the buffering rules. The result of testing the buffering rules can be seen in Fig. 7.

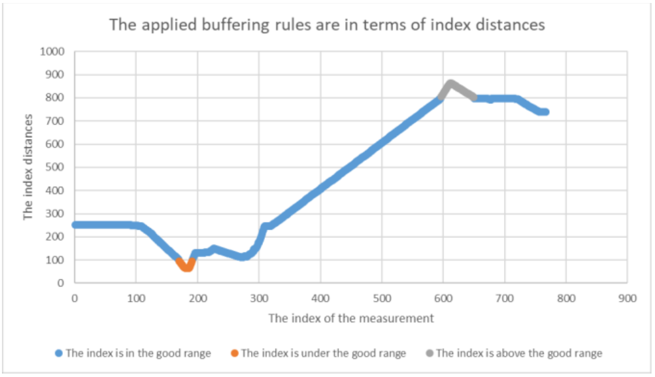

Fig. 7. The result of the testing of the buffering rules.

As shown in Fig. 7, if the difference between the currently displayed frame index from the buffer and the latest frame index received from the server does not fall within the predefined range, the buffering rules will come into effect. We can see the result of the first buffering rule being processed with orange colour in Fig. 7, while the result of the second rule is shown in grey. It can be seen that if the value in question is out of the defined range either upwards or downwards, the rules can restore it to the appropriate range by changing the value of the display attributes.

C. Testing the server-side resource usage

Testing our server software was important for a number of reasons. These included:

- Which hardware component represents the bottleneck in the case of server software when a large number of users are connected?
- What is the resource utilization of the server program for different numbers of users?
- What is the maximum number of users we can serve with an acceptable resolution video stream?

During our research, we continued testing our system by generating virtual users, with more users, and on weaker hardware. In order to be able to test our server-side software solution with hundreds of users, we have created a virtual test environment. In this test environment, there is no need to connect physical users, but the server software is able to create virtual users. With this solution we are able to simulate the presence of actual physical users in the system. During the test, a 3D lab environment was created in this simulated environment, displaying a digitized medical sample consisting of 89 elements. The simulation was set up so that the system would increase the number of connected with a predetermined number. During the test, the following parameters were checked on the server side:

- CPU usage,
- RAM usage,
- GPU usage.

The hardware used to perform the testing had the following specifications:

- 11th Gen Intel(R) Core(TM) i9-11900KF @3.5GHz
- 64 GB DDR4 RAM
- NVIDIA GeForce RTX 2060 SUPER
- Samsung 980 PRO 2TB SSD

The test environment was built in such a way that each user receives 1 image per second from the server at a resolution of 320*180.

TABLE II The results of the server's hardware usage tests.

User Number	Server's CPU Usage (%)	Server's GPU Usage (%)	Server's RAM usage (GB)
5	2.00	91.15	0.208
10	3.21	92.64	0.247
20	6.29	93.26	0.271
30	9.40	94.65	0.297
40	12.96	95.15	0.315
50	16.50	97.32	0.336
60	18.09	98.28	0.367
70	19.26	99.67	0.397
80	20.70	99.86	0.421
90	22.15	100.00	0.447
100	23.82	100.00	0.477

In TABLE II, we can see that with the increase in the number of users, each of the examined hardware resources increases to a different extent. We performed 10 tests for virtual user groups of different numbers, and then averaged the obtained values. The results obtained in this way can be seen in TABLE II. The visualization of the test results can be seen in the following paragraphs.

1) Testing the CPU usage of the server sorftware

In our solution presented in the paper, the CPU unit is not basically responsible for generating the images defined for each user. Nevertheless, in our test we observe that as the number of users increases, CPU usage also increases. This is due to the need to perform several background tasks in generating and serving each user. For example, determining which user should receive image data at the current time moment. The result can be seen in the Fig. 8.

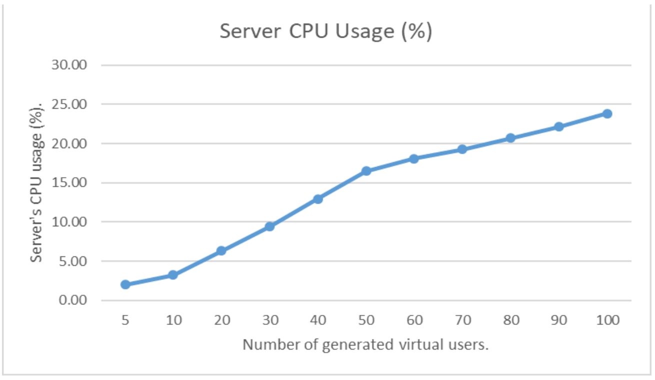

Fig. 8. The test result for the CPU usage of the server software with different number of virtually generated users.

As we can see in Fig. 8, with a higher number of users, the CPU usage of our server also increases. Fig. 8 shows that the CPU utilization of the current version of the server is almost linear but further tests are needed to prove this for sure.

2) Testing the GPU usage of the server software

In our current solution, the most exploited hardware unit is the GPU. This may be because the GPU is responsible for generating the actual image displayed to each user. This may not be a problem with a small number of users, but with close to 100 users we are reaching the limits of our current testing hardware. This can be seen in Fig. 9.

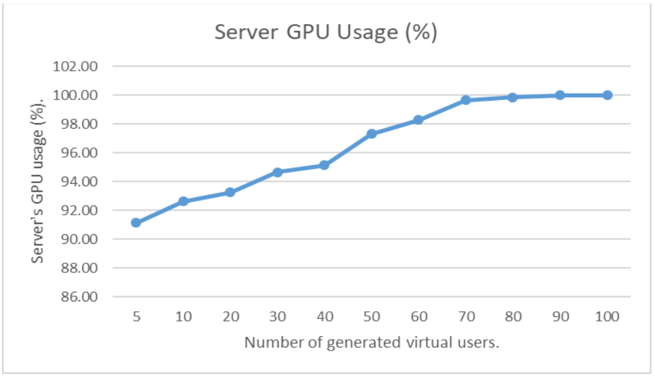

Fig. 9. The test result for the GPU usage of the server software with different number of virtually generated users.

Fig. 9 shows that even with 5 virtual users we are close to the maximum GPU utilization, and with 80 virtual users we reach it.

3) Testing the RAM usage of the server sotfware

Even with up to 100 virtual users, the memory usage of our current solution is low. This can be seen in Fig. 10.

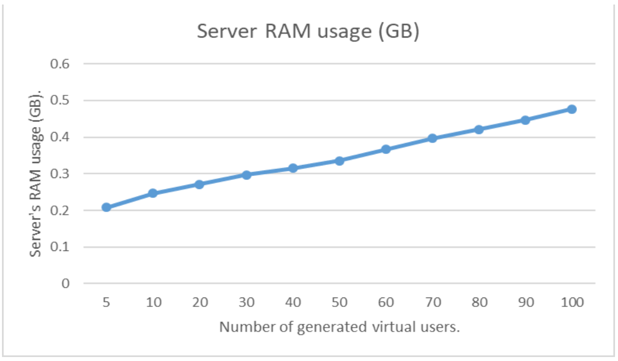

Fig. 10. The test result for the RAM usage of the server software with different number of virtually generated users.

We can see that even with 100 users, we do not reach 1 GB of RAM usage with our server program. The test results show that in our current solution, the RAM usage of the server is not a limiting factor in terms of how many users can connect to our system.

V. Conclusion

In the paper, we presented practical results of our research, which relate to the application of low latency video streaming technology in the field of digital pathology. In the paper, we presented a possible implementation of video streaming technology in the medical field and presented two test cases with which we validated our solution after development. In addition, we presented our latest test results for our server software in the paper. During testing, we created a testing environment in which we are able to simulate a large number of connected users. With this solution, we are able to define the hardware limits of our solution regarding the maximum number of users. Nowadays, the evaluation of 3D digitized pathological samples requires strong hardware resources, but with the client-server solution we have presented, the user can also evaluate pathological serial sections in 3 dimensions on his own, everyday laptop, if a sufficiently powerful server is available to which he can connect.

By applying the solution described in the article and by further developing it, it is possible to eliminate the obstacles related to the hardware capacity of the 3D visualization used in digital pathology.

Acknowledgment

The authors would like to thank the 2019-1.3.1-KK-2019-00007 “Innovációs szolgáltató bázis létrehozása diagnosztikai, terápiás és kutatási célú kiberorvosi rendszerek fejlesztésére” for the financial support, and furthermore we would like to thank AIAM (Applied Informatics and Applied Mathematics) doctoral school of Óbuda University, Budapest, Hungary for their support in this research.

References

[1] P. Baranyi, Á. Csapó, T. Budai, and G. Wersényi, ‘Introducing the Concept of Internet of Digital Reality – Part I’, ACTA POLYTECH HUNG, vol. 18, no. 7, pp. 225–240, 2021, doi: 10.12700/APH.18.7.2021.7.12.

[2] A. Molnár, I. Lovas, and Z. Domozi, ‘Practical Application Possibilities for 3D Models Using Low-resolution Thermal Images’, ACTA POLYTECH HUNG, vol. 18, no. 4, pp. 199–212, 2021, doi: 10.12700/APH.18.4.2021.4.11.

[3] Vincze, M.; Molnar, B.; Kozlovszky, M. 3D Visualization in Digital Medicine Using XR Technology. Future Internet 2023, 15, 284. https://doi.org/10.3390/fi15090284

[4] M. Falk, A. Ynnerman, D. Treanor, and C. Lundstrom, ‘Interactive Visualization of 3D Histopathology in Native Resolution’, IEEE Trans. Visual. Comput. Graphics, vol. 25, no. 1, pp. 1008–1017, Jan. 2019, doi: 10.1109/TVCG.2018.2864816.

[5] R. K. P. Mok, E. W. W. Chan, and R. K. C. Chang, ‘Measuring the quality of experience of HTTP video streaming’, in 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops, Dublin, Ireland, May 2011, pp. 485–492. doi: 10.1109/INM.2011.5990550.

[6] N. Staelens, M. H. Pinson, P. Corriveau, F. D. Turck, and P. Demeester, ‘MEASURING VIDEO QUALITY IN THE NETWORK: FROM QUALITY OF SERVICE TO USER EXPERIENCE’, p. 6.

[7] I. M. I. Zebari, S. R. M. Zeebaree, and H. M. Yasin, ‘Real Time Video Streaming From Multi-Source Using Client-Server for Video Distribution’, in 2019 4th Scientific International Conference Najaf (SICN), Al-Najef, Iraq, Apr. 2019, pp. 109–114. doi: 10.1109/SICN47020.2019.9019347.

[8] S. Park, M. Hoai, A. Bhattacharya, and S. R. Das, ‘Adaptive Streaming of 360-Degree Videos with Reinforcement Learning’, in 2021 IEEE Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, USA, Jan. 2021, pp. 1838–1847. doi: 10.1109/WACV48630.2021.00188.

[9] Liu Yijun, Chen Wenbin, and He Xiaoman, ‘3D graphics engine technology research and implementation’, in 2010 3rd International Conference on Computer Science and Information Technology, Chengdu, China, Jul. 2010, pp. 697–700. doi: 10.1109/ICCSIT.2010.5564684.

[10] A. Thorn, Moving from Unity to Godot: An In-Depth Handbook to Godot for Unity Users. 2020. doi: 10.1007/978-1-4842-5908-5.

[11] Zhao, Y., Luo, C., Tang, C., Chen, D., Codella, N., & Zha, Z. J. (2023). Streaming Video Model. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 14602-14612).

[12] Li, B., Wang, Z., Liu, J., and Zhu, W. 2013. Two decades of Internet video streaming: A retrospective view. ACM Trans. Multimedia Comput. Commun. Appl. 9, 1s, Article 33 (October 2013), 20 pages. DOI: http://dx.doi.org/10.1145/2505805

[13] Yin, G., Jiang, X., Jiang, S., Han, Z., Zheng, N., Yang, H., ... & Qiu, L. (2023). Online Video Streaming Super-Resolution with Adaptive Look-Up Table Fusion. arXiv preprint arXiv:2303.00334.

[14] Dapeng Wu, Y. T. Hou, Wenwu Zhu, Ya-Qin Zhang and J. M. Peha, "Streaming video over the Internet: approaches and directions," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 11, no. 3, pp. 282-300, March 2001, doi: 10.1109/76.911156.

[15] U. Rahamathunnisa and D. R. Saravanan, ‘A SURVEY ON VIDEO STREAMING OVER MULTIMEDIA NETWORKS USING TCP’, . Vol., vol. 53, p. 6, 2005.

[16] B. Li, Z. Wang, J. Liu, and W. Zhu, ‘Two decades of internet video streaming: A retrospective view’, ACM Trans. Multimedia Comput. Commun. Appl., vol. 9, no. 1s, pp. 1–20, Oct. 2013, doi: 10.1145/2505805.

[17] S. Ando, Y. Hayashi, T. Mizuki, and H. Sone, ‘Basic Study on the Method for Real-Time Video Streaming with Low Latency and High Bandwidth Efficiency’, in 2015 IEEE 39th Annual Computer Software and Applications Conference, Taichung, Taiwan, Jul. 2015, pp. 79–82. doi: 10.1109/COMPSAC.2015.217.

[18] M. Michalos, S. Kessanidis, and S. L. Nalmpantis, ‘Dynamic adaptive streaming over HTTP’, Journal of Engineering Science and Technology Review, vol. 5, pp. 30–34, Jun. 2012, doi: 10.25103/jestr.052.06.

[19] Jae Chung and M. Claypool, "Analysis of active queue management," Second IEEE International Symposium on Network Computing and Applications, 2003. NCA 2003., Cambridge, MA, USA, 2003, pp. 359-366, doi: 10.1109/NCA.2003.1201176.

[20] Morais, Wladimir Gonçalves de, Carlos Eduardo Maffini Santos, and Carlos Marcelo Pedroso. ‘Application of Active Queue Management for Real-Time Adaptive Video Streaming’. Telecommunication Systems 79, no. 2 (February 2022): 261–70. https://doi.org/10.1007/s11235-021-00848-0.

[21] Giménez, A.; Murcia, M.A.; Amigó, J.M.; Martínez-Bonastre, O.; Valero, J. New RED-Type TCP-AQM Algorithms Based on Beta Distribution Drop Functions. Appl. Sci. 2022, 12, 11176. https://doi.org/10.3390/app122111176

[22] Yue-Dong Xu, Zhen-Yu Wang and Hua Wang, "ARED: a novel adaptive congestion controller," 2005 International Conference on Machine Learning and Cybernetics, Guangzhou, China, 2005, pp. 708-714 Vol. 2, doi: 10.1109/ICMLC.2005.1527036.

[23] D. M. Raghuvanshi, B. Annappa and M. P. Tahiliani, "On the Effectiveness of CoDel for Active Queue Management," 2013 Third International Conference on Advanced Computing and Communication Technologies (ACCT), Rohtak, India, 2013, pp. 107-114, doi: 10.1109/ACCT.2013.27.

[24] I. Järvinen and M. Kojo, "Evaluating CoDel, PIE, and HRED AQM techniques with load transients," 39th Annual IEEE Conference on Local Computer Networks, Edmonton, AB, Canada, 2014, pp. 159-167, doi: 10.1109/LCN.2014.6925768.

[25] H. H. Soliman, H. M. El-Bakry, and M. Reda, ‘Real-Time Transmission of Video Streaming over Computer Networks’, p. 13.

[26] Davis CR, Rosenfield LK. Looking at plastic surgery through google glass: Part 1. Systematic review of google glass evidence and the first plastic surgical procedures. Plast Reconstr Surg. 2015;135:918–928

[27] Yoon, Hyoseok, Sun Kyung Kim, Youngho Lee, and Jongmyung Choi. ‘Google Glass-Supported Cooperative Training for Health Professionals: A Case Study Based on Using Remote Desktop Virtual Support’. Journal of Multidisciplinary Healthcare Volume 14 (June 2021): 1451–62. https://doi.org/10.2147/JMDH.S311766.

[28] A. Schneider et al., ‘Wireless live streaming video of surgical operations: an evaluation of communication quality’, J Telemed Telecare, vol. 13, no. 8, pp. 391–396, Dec. 2007, doi: 10.1258/135763307783064386.

[29] M. M. Jack, D. A. Gattozzi, P. J. Camarata, and K. J. Shah, ‘Live-Streaming Surgery for Medical Student Education - Educational Solutions in Neurosurgery During the COVID-19 Pandemic’, Journal of Surgical Education, vol. 78, no. 1, pp. 99–103, Jan. 2021, doi: 10.1016/j.jsurg.2020.07.005.

[30] Hanna, Matthew G., Ishtiaque Ahmed, Jeffrey Nine, Shyam Prajapati, and Liron Pantanowitz. ‘Augmented Reality Technology Using Microsoft HoloLens in Anatomic Pathology’. Archives of Pathology & Laboratory Medicine 142, no. 5 (1 May 2018): 638–44. https://doi.org/10.5858/arpa.2017-0189-OA.

[31] Vitkovski, Taisia, Tawfiqul Bhuiya, and Michael Esposito. ‘Utility of Telepathology as a Consultation Tool between an Off-Site Surgical Pathology Suite and Affiliated Hospitals in the Frozen Section Diagnosis of Lung Neoplasms’. Journal of Pathology Informatics 6, no. 1 (January 2015): 55. https://doi.org/10.4103/2153-3539.168515.

[32] M. Y. Fuller, S. Mukhopadhyay, and J. M. Gardner, ‘Using the Periscope Live Video-Streaming Application for Global Pathology Education: A Brief Introduction’, Archives of Pathology & Laboratory Medicine, vol. 140, no. 11, pp. 1273–1280, Nov. 2016, doi: 10.5858/arpa.2016-0268-SA.

[33] M. Vincze, B. Biricz, M. Kozlovszky and A. Benhamida, "Real-time video streaming in medicine using virtual reality," 2023 IEEE 17th International Symposium on Applied Computational Intelligence and Informatics (SACI), Timisoara, Romania, 2023, pp. 000183-000188, doi: 10.1109/SACI58269.2023.10158657.