INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2025, VOL. 71, NO. 4, PP. 1-6
Manuscript received June 02, 2025; revised September 2025. doi: 10.24425/ijet.2025.155460

3D ultrasound reconstruction based on free hand acquisitions with motion estimation

Paweł Żogał, Krzysztof J. Opieliński, and Andrzej Wiktorowicz

***Abstract*—We present a method for motion artifacts reduction during 3D volume reconstruction of free hand 2D ultrasound sequences. Motion estimation and additional filtering improves quality of free hand 3D ultrasound data resulting in improved imaging. Reconstructed 3D data was visualized using the OpenGL ES framework and GLSL shader language allowing real-time rendering on an embedded class GPU device.**

***Keywords*—3D ultrasound; image stabilization; motion estimation; volume rendering**

I. Introduction

ULTRASOUND imaging is a safe and relatively cheap diagnostic modality. 3D reconstruction and visualization may constitute an additional value to a regular examination workflow. The most common application is OB/GYN for fetal diagnostics [1]. Other use cases are also cardiology [1], surgical navigation [2] or vascular imaging. This modality requires that a system is able to collect 3D volumetric data. It can be realized using either: a 2D array probe, a 1D mechanical swing array probe, a 1D array probe with a tracking device or a regular 1D array probe with assumed movement trajectory [3]. Each technique is a trade-off between data quality and system complexity and price. The latter method called also the "free-hand 3D ultrasound" technique requires least technical resources but is also definitely more prone to reconstruction errors.

II. Objectives

The presented work has been conducted within a NCBiR research project no. POIR.01.01.01-00-1462/19 with the goal of development a high channel ultraportable ultrasound scanner for veterinary and human medicine applications. One of the system components was 3D imaging mode and given the project requirements and assumptions the free-hand approach was the only option, thus we were not able to use any kind of probe tracking technology, like optical, inertial or optical tracking. Within our work we developed 3D volume reconstruction based on following predefined ultrasound probe movement trajectories: linear, swing and rotational [3]. The objective was to deliver image quality comparable with high class standalone devices. To improve the reconstruction quality we developed an algorithm for image stabilization and motion estimation. Next steps were data filtering and visualization. Embedded system performance had to be considered, still allowing for real-time data rendering.

III. Methods

A. Motion model

In this work we assumed a linear probe motion model. The imaging plane is denoted by X and Y axes and the principal movement is along the perpendicular Z axis. There can be slight shifts and rotations in the XY plane and the movement along the Z axis covers a certain distance with inconsistent speed. The recorded frame sequence is denoted as:

$$\boldsymbol{I} = \{ I_1, I_2, \ldots, I_k, \ldots, I_N \} \tag{1}$$

where $k = 1 \ldots N$ is the frame index. The transformation between two consecutive frames can be expressed by a affine transformation matrix consisting of three transforms: rotation along Z axis, shift in the XY plane and another shift along the Z axis:

$$M(k, k+1) = T(0,0,dz) * T(dx, dy, 0) * Rz(\alpha) =$$

$$\begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & dx \\ \sin(\alpha) & \cos(\alpha) & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Our model constrains the movement from 6 degrees of freedom to 4: one rotation and 3 shifts.

B. Motion Estimation

Motion estimation algorithm aims at calculating $[\alpha, dx, dy, dz]$ estimates. Having a series of affine transforms between the recorded frames one can create a volumetric dataset and map each voxel to a specific pixel. In our case we split the process into 2D motion estimation where parameters $[\alpha, dx, dy]$ are calculated and linear movement speed approximation where $[dz]$ is determined.

2D transformation calculation is based on patch matching. A patch set is defined as:

$$\boldsymbol{P} = \{ P_1, P_2, \ldots, P_j, \ldots, P_M \} \tag{3}$$

where $j = 1 .. M$ is the patch index. Each patch is a rectangular region centered around a certain pixel in the source image. For each two consecutive frames the algorithm tries to find optimal

The research described in the paper was carried out as part of the POIR.01.01.01-00-1462/19 research project funded by the National Centre for Research and Development (NCBR) with funds from the European Union.

Paweł Żogał is with CYBERMEDICS, Wrocław, Poland (e-mail: pzogal@cybermedics.pl).

Krzysztof J. Opieliński is with Department of Acoustics, Multimedia and Signal Processing, Faculty of Electronics, Photonics and Microsystems, Wroclaw University of Science and Technology, Wrocław (e-mail: krzysztof.opielinski@pwr.edu.pl).

Andrzej Wiktorowicz is with DRAMIŃSKI S.A., Poland (e-mail: andrzej.wiktorowicz@draminski.com).

2D transforms between all patches. The transform is defined as a combination of rotation around the middle point of the patch and some shift by a 2D vector: $T(\alpha, dx, dy)$. The optimization algorithm searches for the highest similarity measure between the transformed patch P_j^{k+1} in the image I^{k+1} and the original patch P_j^k given parameters: $[\alpha_j^{k+1}, dx_j^{k+1}, dy_j^{k+1}]$. Similarity measure used is Pearson correlation coefficient given by following equation:

$$r(I_1, I_2) = \frac{\sum_{i=1}^{N}(I_1(X_i)-\bar{I_1})(I_2(Y_i)-\bar{I_2})}{\sqrt{\sum_{i=1}^{N}(I_1(X_i)-\bar{I_1})^2}\sqrt{\sum_{i=1}^{N}(I_2(Y_i)-\bar{I_2})^2}} \quad (4)$$

where:

- I_1, I_2 are two image fragments, each consisting of N pixels
- $I_1(X_i)$ is the pixel value at the position $X_i=[x_i,y_i,0]^T$
- Y_i is the position X_i transformed by $T(\alpha, dx, dy)$

In this work we used Nelder-Mead optimization algorithm [4] to find transformation parameters for each patch pair. The Pearson correlation coefficient is also called normalized cross correlation (NCC), where value 1 means total correlation, 0 – no correlation and -1 negative correlation. We assumed that if NCC is below certain threshold the patches don't match. Below, there is an example of two frames with patch designation and matching result:

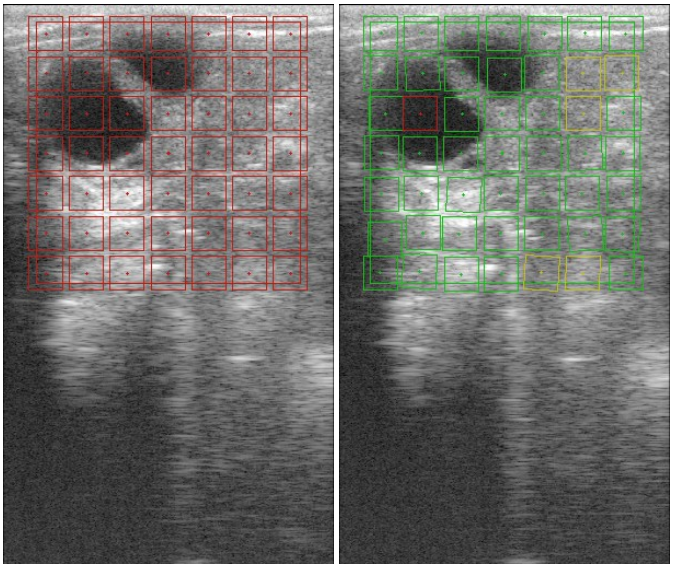

Fig. 1. 2D patch matching, left: base image with rectangular patches, right: following image with transformed patches.

Patches that have good match (high NCC value) are marked as green. Yellow denotes middle NCC range and red marks cases where no good match could be found. The latter could happen, for example, in cases where there is not enough value variation in the patch region. Total image transformation is then calculated as an average of the single patches transformations. After this step values for $[\alpha, dx, dy]$ in (2) are fixed and the next stage determines the last parameter: dz.

Perpendicular ultrasound probe image based shift estimation was explored in several work groups, starting around 1996 [5],[6],[7],[8],[9],[10],[11]. All mentioned contributions are based on one phenomenon [5]: movement along the Z axis decreases the NCC in the regions where uncompressed amplitude follows the Rayleigh distribution. This can be expressed by following equation:

$$r(I_1, I_2) = \lambda^2 = e^{-\frac{dz^2}{k^2}} \quad (5)$$

where dz is the shift value and k is some constant coefficient dependent on the probe and the distance from the probe surface. One could of course try to calibrate the value for different probes and depths, but it was beyond the scope of our project. Another uncertainty is the direction of the movement, equation (5) gives only the absolute value of dz. We assumed that the probe moves only in one direction and the total distance travelled within the recorded sequence is set as a parameter ΔZ:

$$\Delta Z = \sum_{i=2}^{N} dz_i \quad (6)$$

where:

- ΔZ is the total distance travelled by the probe (for example: 5, 10 or 15cm)
- dz_i is the shift along the Z axis between the i-th frame and the previous frame

Given (5) we can write:

$$dz_i = k * \sqrt{\log\left(r(I_i, I_{i-1})\right)} \quad (7)$$

Combining (6) and (7) we can get following equation:

$$dz_i = \frac{\Delta Z \sqrt{\log\left(r(I_i, I_{i-1})\right)}}{\sum_{i=2}^{N}\sqrt{log\left(r(I_i, I_{i-1})\right)}}, \quad (8)$$

C. Volume Reconstruction

Having determined 3D affine transforms between all consecutive 2D ultrasound frames one could reconstruct a three dimensional grid of a volumetric dataset. For each voxel coordinates $P=[x,y,z]^T$ one can find two closest 2D frames based on Z coordinates. Next, XY projections are calculated and the final voxel value is interpolated between the two pixel values.

D. Data Filtering

Next stage in our processing pipeline is data filtering. Source images visible in Fig.1 are noisy and wouldn't produce smooth surfaces in 3D visualization. We aimed at choosing a 3D filter producing smooth surfaces while preserving structure details. Within our work we tested 5 filters: gaussian low-pass filter, median filter, Kuwahara filter [12] with its adaptive version [13] and finally lowest variance filter [14].

The simplest filter is gaussian blur. It is realized by convolving volumetric dataset expressed as a 3D array by respective gaussian kernel. The advantages are: good noise suppression and easy implementation. On the other hand one looses small details and the edges get blurred, too.

Another classical algorithm implemented in our project was median filter. The principle is that for each voxel V_i there is a defined neighborhood Ω_i (for example 3x3x3 or 5x5x5) and the filter output is the median value of all values in Ω_i. A typical feature of median filters is the ability to reduce noise while preserving the edges.

Next the Kuwahara [12] filter was analyzed. It divides the voxel neighborhood into 8 overlapping sub-cubes. Given the

neighborhood size $(2*N-1)^3$ each sub-cube has size equal to N^3. Central voxel is the only voxel present in all sub-cubes. The sub-cube with the least variance is chosen and its average value replaces the central voxel value. In principle, the filter is designed to preserve structure details and reduce noise. We also tested the adaptive version of the Kuwahara filter [13]. In this case the sub-cubes with sizes from 2x2x2 to MxMxM are tested and the one with the least variance is chosen for mean value calculation like in the original algorithm.

Finally, the lowest variance path filter [14] was implemented. In the first stage for each voxel a vector of lowest variance is determined. It's done by sampling unitary sphere and for each vector 3D data is traversed from the given voxel in particular direction and its reverse. N samples are collected and their variance is calculated. The vector that produces the lowest variance is stored in an auxiliary 3D dataset. Unitary sphere sampling vectors are given by equation (9):

$$V(i,j) = \begin{pmatrix} \cos(\theta_i) & 0 & -sin(\theta_i) \\ 0 & 1 & 0 \\ sin(\theta_i) & 0 & cos(\theta_i) \end{pmatrix} * \begin{pmatrix} cos(\varphi_j) & -sin(\varphi_j) & 0 \\ sin(\varphi_j) & cos(\varphi_j) & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad (9)$$

where: $\theta_i = \frac{2\pi * i}{i_{max}}$, $\varphi_j = \frac{\pi * j}{j_{max}}$

The second stage is traversing the volume for each voxel and going along consecutive lowest-variance vectors in both positive and negative directions. Number of steps is fixed and the final output value is the average of collected samples. The principle of the algorithm is to average voxel values along a path that has similar values. The method produces very appealing results, but requires much more computing resources than other techniques tested within our work.

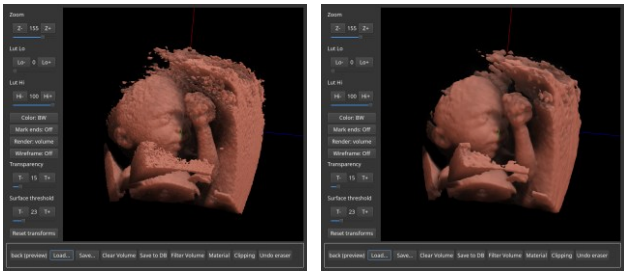

Fig. 2. Left: no filtering, right: gaussian blur

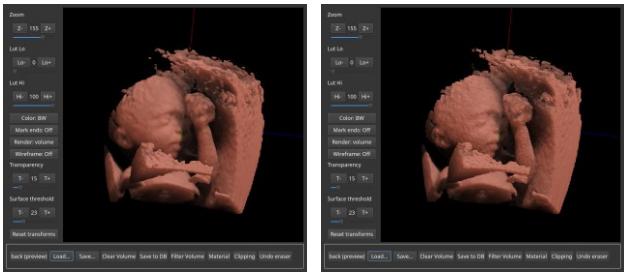

Fig. 3. Left: median filter, right: Kuwahara filter

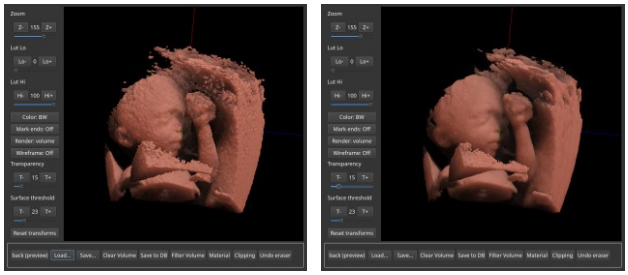

Fig. 4. Left: adaptive Kuwahara filter, right: lowest variance path filter

E. 3D Visualization

Finally the data was visualized by an algorithm of choice. Within the project we implemented three volume rendering methods: maximum intensity projection (MIP), semi-transparent rendering and surface rendering. The algorithms were implemented using OpenGL shader language and adjusted to available GPU to reach real-time objective. If needed, the pixel values were inverted which in particular case helped visualizing follicles that appear as black holes on ultrasound images whereas we wanted to see them as solid structures to asses visually their morphology and determine the total follicle count in the acquired sequence.

For all rendering modes ray-casting [15],[16] method was used. The volume is defined as a cuboid consisting of 8 vertices having appropriate texture coordinates that refer to the voxel coordinates of the volumetric 3D array. In the OpenGL rendering pipeline vertex shaders are used for geometric transformation and fragment shaders are responsible for calculating the final color of a pixel. Fragment shaders contain a loop where a ray through the volume is traversed and the coordinates are taken from the interpolated texture coordinates.

In the MIP mode, at each traverse step, the volume voxel coordinates are calculated and the voxel value is mapped to intensity function. The final output value is the maximum calculated along the particular ray.

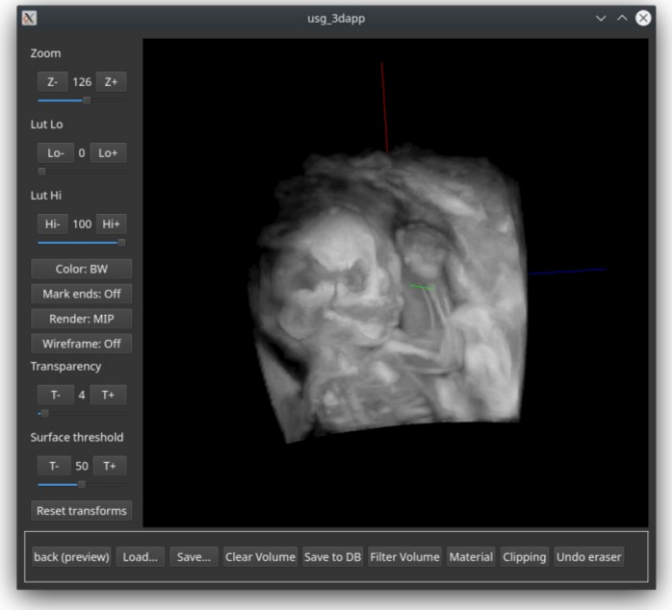

Fig. 5. 3D volume model of a human fetus: maximum intensity projection.

In the semi-transparent mode, the voxel intensity is mapped to a transparency function and at each step initial light value is

reduced by the mapped amount. The loop finishes when the ray point is either out of volume or the intensity goes below certain threshold.

Fig. 6. 3D volume model of a human fetus: semi-transparent rendering.

Finally, during surface rendering, the voxel value is also mapped to a transparency value, but the mapping function is very steep in some range. This leads to a fast stopping – determining when a surface point has been hit. Next, a surface normal vector is approximated by a 3D gradient vector and the final color is calculated using the Blinn-Phong model [17].

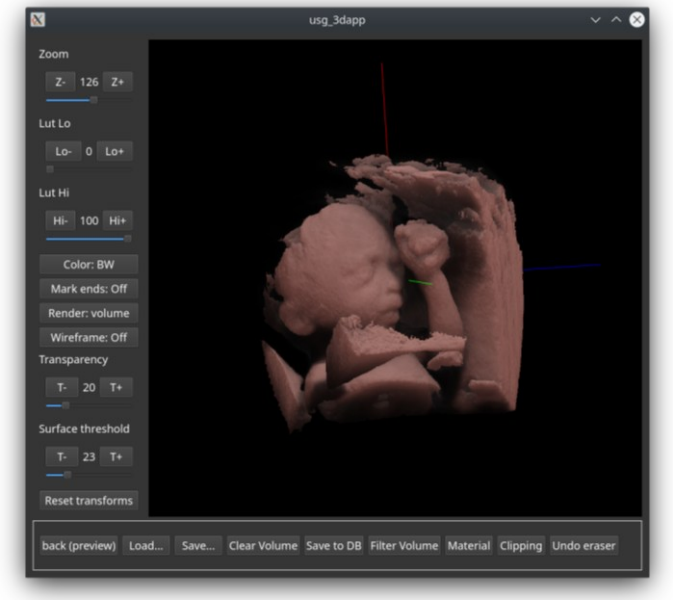

Fig. 7. 3D volume model of a human fetus: surface rendering.

IV. Results

The test data used were 2D sequences of bovine ovary recordings. 3D surface rendering was used to visualize ovarian follicles. 3D imaging allows for fast follicle count and morphology assessment. Below some sample cross-sectional and 3D images are shown.

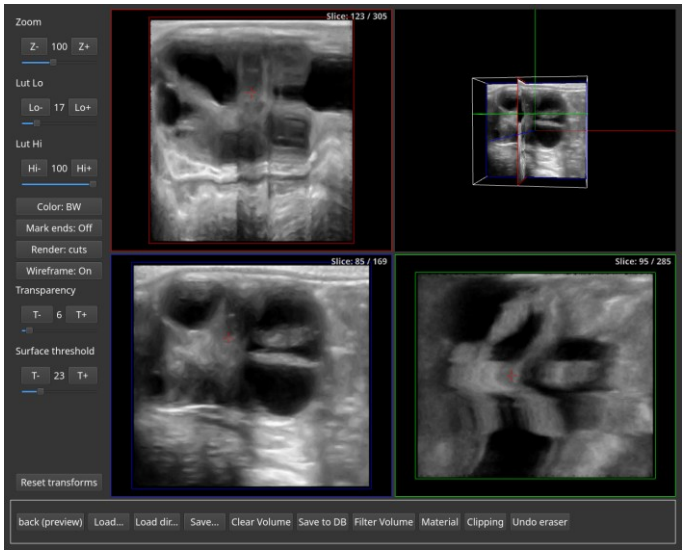

Fig. 8. 2D cross-sections of the 3D volume, no motion correction.

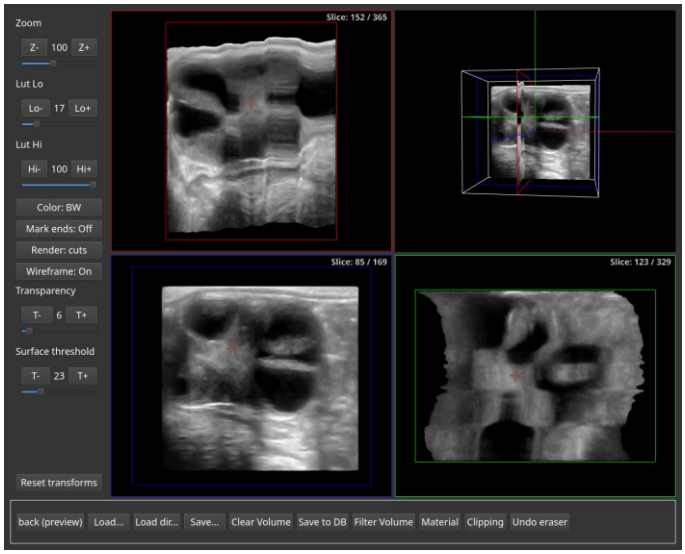

Fig. 9. 2D cross-sections of the 3D volume, motion estimation in imaging plane only.

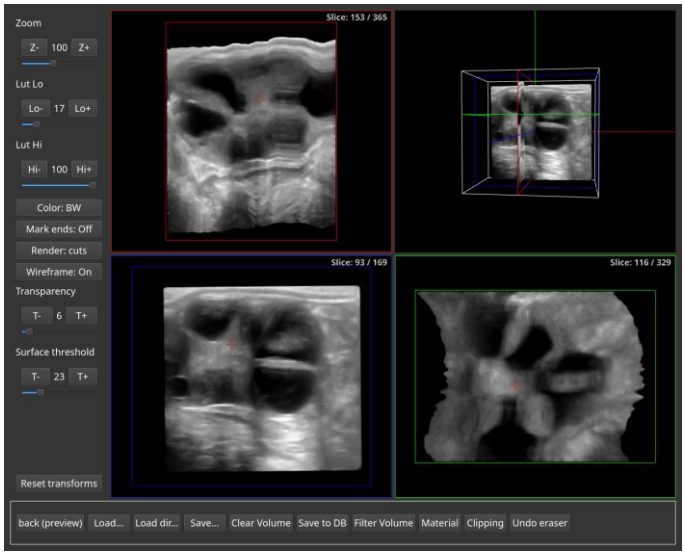

Fig. 10. 2D cross-sections of the 3D volume, motion estimation in 3 axes.

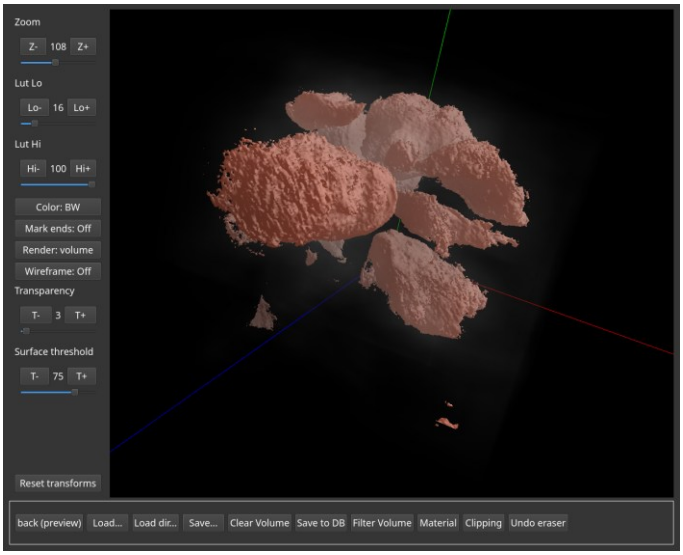

Fig. 11. 3D surface rendering of the follicles, no motion correction.

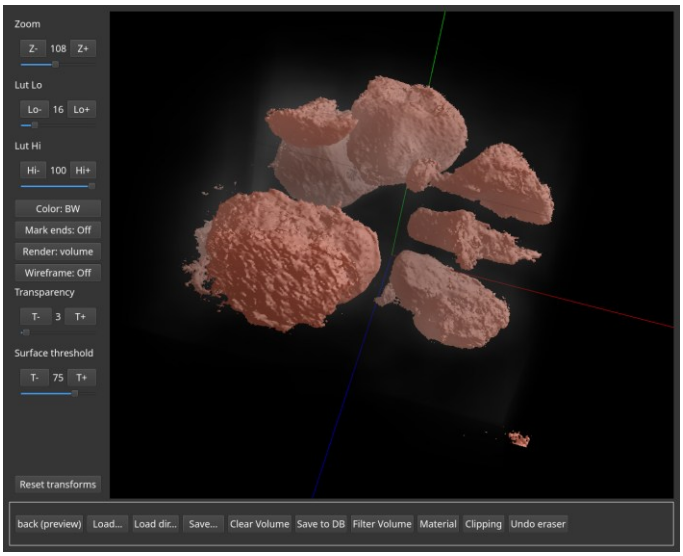

Fig. 12. 3D surface rendering of the follicles, only motion correction, no filtering.

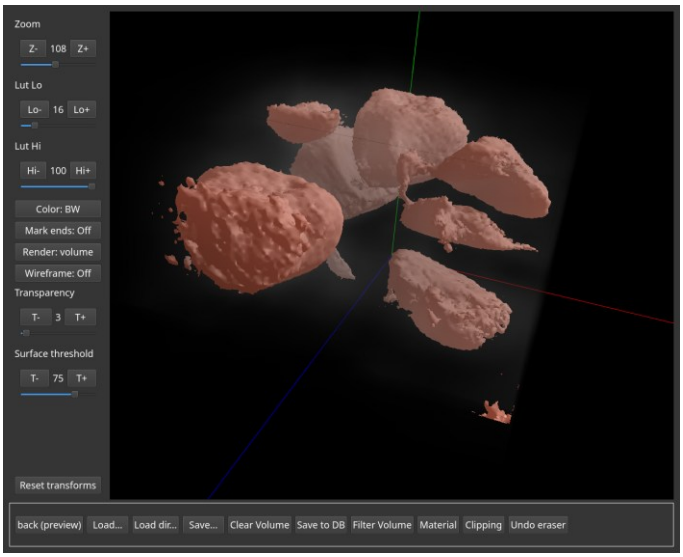

Fig. 13. 3D surface rendering of the follicles, motion correction with lowest-variance filtering.

V. Discussion

We could show that even with limited technical resources one can improve the quality of 3D volume reconstruction. The word quality here is a descriptive term, we cannot measure it with numbers, but it can be more or less appealing to the user. Using freehand technique there are several limitations in term of reconstruction accuracy. We can only estimate movement speed with an unknown scaling factor. In the end it's up to the user to decide how much one needs to stretch the volumetric dataset in the 3rd dimension. The same algorithm can be however combined also with more advanced tracking techniques like inertial, optical or electromagnetic tracking. This could be a potential path for the continuation of the presented work. Another possible direction to extend this research would be finding ways how to utilize deep learning techniques to perform motion estimation or correction given aforementioned sources of probe localization. Regarding 3D data visualization techniques, the constant progress in GPU technologies will definitely allow using more elaborate algorithms with even more appealing results and less energy consumption. Additionally, offloading heavy calculations to the GPU could noticeably improve algorithm performance, leading to a quasi-real time reconstruction.

VI. Conclusion

Motion estimation algorithm improves imaging quality by reducing out of trajectory movement artifacts. OpenGL GPU optimized programming allows real-time imaging and thus leading to quicker and more accurate assessments of the ultrasound exams. Our work shows that there's potential to improve 3D reconstruction and 3D imaging on portable ultrasound devices. The constraints are clear, on one hand one demands as much computing power as possible on another hand the portability requires low energy designs and also the costs have to be feasible for the final customers. We hope that constantly improving GPU technology will allow implementing more and more demanding algorithms on low-powered embedded devices while keeping the prices affordable. Recent advances in edge AI computing also look very promising. And it's another area to explore in terms of motion estimation and data filtering.

References

[1] Q. Huang, Z. Zeng, "A Review on Real-Time 3D Ultrasound Imaging Technology", Biomed Research International, 2017:2017:6027029, 2017. https://doi.org/10.1155/2017/6027029

[2] P. Zogal, G. Sakas, W. Rösch, D. Baltas, "BiopSee® - transperineal stereotactic navigated prostate biopsy", Journal of Contemporary Brachytherapy, vol. 3, pp. 2:91-95, 2011. https://doi.org/10.5114/jcb.2011.23203

[3] A. Fenster, G. Parraga, J. Bax "Three-dimensional ultrasound scanning", Interface Focus, vol. 1(4), pp. 503-19, 2011. https://doi.org/10.1098/rsfs.2011.0019

[4] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, „Numerical Recipes in C. The Art of Scientific Computing", 2nd Edition, Cambridge University Press, New York, 992.

[5] J.F. Chen, J. Fowlkes, P. Carson, J. Rubin, "Determination of scan-plane motion using speckle decorrelation: Theoretical considerations and initial test", International Journal of Imaging Systems and Technology, vol. 8(1), pp. 38-44, 1997. https://doi.org/10.1002/(SICI)1098-1098(1997)8:1<38::AID-IMA5>3.0.CO;2-U

[6] T.A. Tuthill, J.F. Krücker, J.B. Fowlkes, P.L. Carson, "Automated three-dimensional US frame positioning computed from elevational speckle decorrelation", Radiology, vol. 209(2), pp. 575-582, 1998. https://doi.org/10.1148/radiology.209.2.9807593

[7] R.F. Chang, W.J. Wu, D.R. Chen, W.M. Chen, W. Shu, J.H. Lee, L.B. Jeng, "3-D US frame positioning using speckle decorrelation and image registration", Ultrasound in Medicine and Biology, vo. 29(6), pp. 801-812, 2003. https://doi.org/10.1016/s0301-5629(03)00036-x

[8] W. Smith, A. Fenster, "Statistical analysis of decorrelation-based transducer tracking for three-dimensional ultrasound", Medical Physysics, vol. 30(7), pp. 1580-1591, 2003. https://doi.org/10.1118/1.1577231

[9] A.H. Gee, J.R. Housden, P. Hassenpflug, G.M. Treece, R.W. Prager, "Sensorless free-hand 3D ultrasound in real tissue: speckle decorrelation without fully developed speckle", Medical Image Analysis, vol. 10(2), pp. 137-149, 2006. https://doi.org/10.1016/j.media.2005.08.001

[10] R.J. Housden, A.H. Gee, G.M. Treece, R.W. Prager, "Sensorless reconstruction of uncon-strained freehand 3D ultrasound data" Ultrasound in Medicine and Biology, vol. 33(3), pp. 408-419, 2007. https://doi.org/10.1016/j.ultrasmedbio.2006.09.015

[11] C. Laporte, T. Arbel, "Probabilistic speckle decorrelation for 3D ultrasound", Medical Image Computing and Computer-Assisted Intervention, vol. 10(Pt 1), pp. 925-932, 2007. https://doi.org/10.1007/978-3-540-75757-3_112

[12] M. Kuwahara. K. Hachimura, S. Eiho, M. Kinoshita, "Processing of RI-angiocardiographic images," in Digital Processing of Biomedical Images, K. Preston Jr. and M. Onoe (Editors), New York: Plenum, pp. 187–202, 1976. https://doi.org/10.1007/978-1-4684-0769-3_13

[13] K. Bartyzel, "Adaptative Kuwahara filter", Signal, Image and Video Processing, vol. 10, pp. 663-670, 2015. https://doi.org/10.1007/s11760-015-0791-3

[14] V. Solteszova, L.E.S. Helljesen, W. Wein, O.H. Gilja, I. Viola, "Lowest-Variance Streamlines for Filtering of 3D Ultrasound", in Eurographics Workshop on Visual Computing for Biology and Medicine, The Eurographics Association, 2012. https://doi.org/10.2312/VCBM/VCBM12/041-048

[15] F. Randima, "GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics", Pearson Higher Education, 2004.

[16] S.D. Roth, "Ray Casting for Modeling Solids", Computer Graphics and Image Processing", vol. 18(2), pp. 109-144, 1982. https://doi.org/10.1016/0146-664X(82)90169-1

[17] J. Blinn, "Models of light reflection for computer synthesized pictures" in Proc. of International Conference on Computer Graphics and Interactive Techniques, pp. 192-198, 1977. https://doi.org/10.1145/563858.563893