User-position aware adaptive display of 3D data without additional stereoscopic hardware

Udo Birk[1,2], Philipp Roebrock[1]
[1] Institut für Photonics und ICT, Hochschule für Technik und Wirtschaft HTW Chur, Switzerland;
[2] Institut für Physik, Universität Mainz, Germany

ABSTRACT

Stereoscopic vision modules have seen limited success in both engineering and consumer world, due to the required additional hardware (image acquisition, Virtual Reality headsets, 3D glasses). In the last years, especially the gaming and education sectors have benefited from such specialized headgear, providing virtual or augmented reality. However, many other industrial and biomedical applications such as e.g. computer aided design (CAD) or tomographic data display, so far have not fully exploited the increased 3D rendering capabilities of present-day computer hardware.

We present an approach to use standard desktop PC hardware (monitor and webcam) to display user-position aware projections of 3D data without additional headgear. The user position is detected from webcam images, and the rendered 3D data (i.e. the view) is adjusted to match the corresponding user position, resulting in a quasi virtual reality rendering, albeit without the 3D effect of proper 3D head-gear. The approach has many applications from medical imaging, to construction and CAD, to architecture, to exhibitions, arts and performances.

Depending on the user location, i.e. the detected head position, the data is rendered differently to attribute for the user view angle (zoom) and direction. As the user moves his or her head in front of the monitor, different features of the rendered object become visible. As the user moves closer to the screen, the view angle of the rendered data is decreased, resulting in a zoomed-in version of the rendered object.

Keywords: 3D rendering, virtual reality, user position detection.

1. INTRODUCTION

Many vision applications today rely on the rendering of 3D data. Although modern computer hardware permits the rendering of such 3D data in a virtual reality (VR) or an augmented reality (AR) environment[1], such hardware is often provided in the form of a head gear, such as specialized glasses or position indicators[2]. Today, the presence of such VR or AR glasses is still limited. To a large degree this is because the glasses themselves are not suitable (because of a dirty, harsh, or wet environment unsuitable for electronic equipment), not available (because they are expensive), or not accepted (because they are uncomfortable or even potentially dangerous[3]). For example, it could be shown that the use of some VR headgear lead to a temporary loss of binocular vision[4,5]. In addition, the rendering needs to be fast in order to allow for a realistic movement of the object and for a smooth transition when updating the view and/or the zoom. Other real-time adapted rendering techniques often make use of a two pass algorithm; first coarsely estimate the viewing direction in the first pass, and search for a better match in the neighborhood of the previously found positions in the second pass[6]. However, such approaches can be computationally complex and thus time consuming. In our approach, we present a single pass user position extraction and subsequent rendering algorithm.

As another alternative to Virtual Reality, so-called “immersive” displays have been employed to present the user with 3D rendering of data[7]. However, also this approach suffers from some drawback: For example, it could be shown that the design of the data presentation (2D or 3D) may have more impact than the performance of the 3D rendering system can potentially compensate.[8] If “immersion” is a requirement for the application, it is important to take both task analysis and engineering requirements into consideration in order to formulate specifications for the display systems, as well as to understand the engineering tradeoffs and human factors issues involved.[9]

Fast rendering of 3D datasets is especially useful in the context of computer aided design (CAD). The design cycle of a CAD developed device has two main phases. During the initial design of the instrument, a precise but static 3D model of the various components is developed. Then comes the test phase of the integrated system, which is integrated in modern CAD software, making use of scientific methods and experiments. Many devices have moving parts, which are partially

hidden behind other components or housing parts. All of these parts can lead to collisions during operation, limit the travel range or otherwise cause failures. To avoid accidents resulting from interactions of parts, the movements of the instrument parts during operation must be carefully tested in a pre-experiment phase, which allows us to detect possible interference. As it can sometimes be difficult to foresee all of the possible configurations of a multi-part CAD constructed device, especially when the device itself is to be embedded in a larger system, it is sometimes advisable to have a human supervised virtual test phase of the device. This is one possible application of our approach for user-position aware fast 3D rendering.

Several goals were realized in this project: A viewer application was designed and realized for fast rendering of 3D datasets. The viewer application was implemented in Matlab (The Mathworks Inc., Natick, MA). In order to test the suitability of our approach in a realistic, real-time rendering application, a variety of different 3D models were used. An interface was provided to integrate and load custom 3D datasets. One goal of the project was to adapt available 3D models (3D datasets) and to convert these models into a format that will be loaded and animated in the dedicated viewer application. In addition, webcam control is added to the viewer application, and linked to the rendering engine. The interface between the extracted user position information and the 3D rendering part provides access to the view angle (zoom) and direction.

After the implementation phase the approach was tested for performance. 3D datasets (meshes) with varying number of triangles were used to estimate the impact of the size of the dataset on the performance. Likewise, camera settings and settings of the Viola Jones face detection algorithm[10] used for the extraction of the user head position were tested with respect to their impact on the performance of the VR rendering.

2. PRINCIPLE

Figure 1 shows an example of a rendered 3D object used during the performance tests. The 3D object shown is the surface of the moon. This dataset was made available from NASA[11], and is the biggest dataset used in our tests. The 3D dataset underlying this object consists of 750 000 vertices, i.e. 250 000 individual surface patches. For realistic setting, one or more light sources are added to the rendering process, in order to provide shadows, and give highlight to surface structures as indicated in Figure 1.

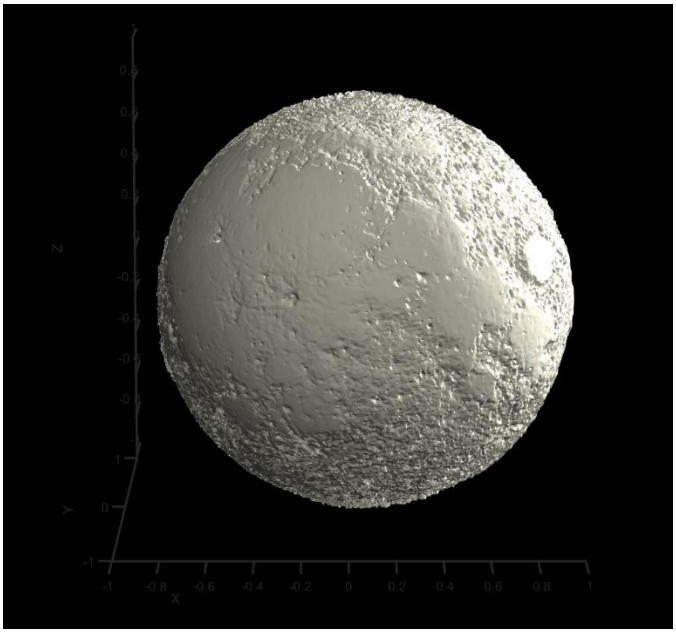

Figure 1. 3D rendering of a CAD model of the lunar surface[11].

Figure 2 illustrates the operation principle of our approach, which is straightforward: 1) From webcam images, the head position (red line) and the head size (blue square) of a single user is extracted and tracked. 2) From the head position data, the tilt angle (viewing direction or camera position) of the rendered data is adjusted. 3) From the head size, the

distance between the user and the screen is estimated; this estimate is used to adjust the view angle (camera zoom) of the rendered data, effectively changing the field of view presented to the user. When the user moves in front of the webcam, different parts of the object are revealed on the screen corresponding to the different viewing directions of the user.

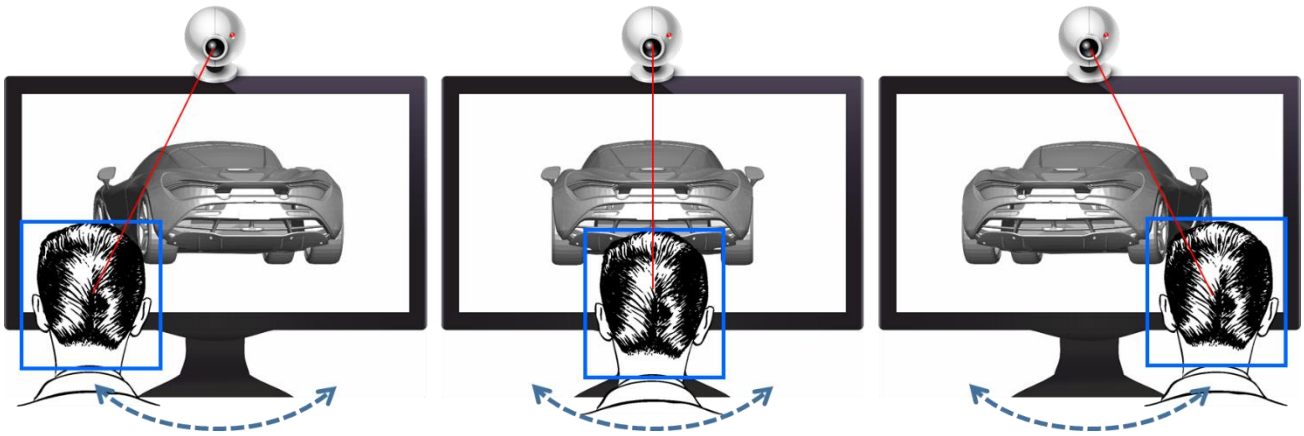

Figure 2. Principle of the user-position aware 3D rendering. Depending on the detected user head position, the rendering is adjusted to attribute for the user view direction and field of view (zoom). As the user moves to the left or right, different features of the rendered object become visible. As the user moves closer to the screen, the view angle decreases, resulting in a zoomed-in version of the rendered object (see below). Image parts: Wiki commons.

In our every-day experience, as we move closer to an object, the extent of the scene which we observe appears to be decreased and the object appears larger. In the viewer application, this experience is mimicked by zooming into the object. For a 3D dataset, this is implemented by decreasing the camera view angle in the Matlab viewer application. Figure 3 illustrates the operation principle of this approach. In this screenshot of the rendered 3D model of a car (shown in Figure 2), the object is zoomed in, detailing on the right rear lights.

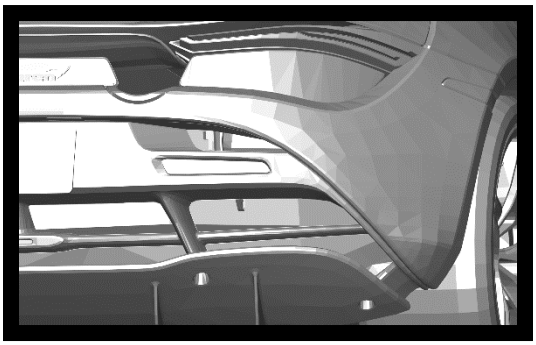

Figure 3. Close up view of the right rear lights of the rendered car.

3. IMPLEMENTATION

The viewer application is implemented using Matlab 2017a. Performance tests are carried out on a dual-core (2 x 2.6 GHz) Windows PC (Intel i5-7300U) with hardware accelerated rendering supported by the processor graphics card (Intel HD Graphics 620). The camera exposure time is fixed to 2^{-9} s in order to avoid variability of the performance due to changes in the lighting conditions. Likewise the automatic white balance of the webcam is switched off, in order to optimize performance of the Viola Jones feature extraction algorithm used to detect the user's face in the webcam image.

The face detection is implemented as Haar feature-based cascade classifiers, using the OpenCV face detection training set[12]. From the resulting bounding box around the face feature, the center is extracted and treated as the value of the head position. The area of the bounding box is treated as the size of the head. Intrinsic variations of the face detection process lead to fluctuations of the detected bounding box size, although without significant effect on the center of the bounding box. An infinite impulse response (IIR) filter with filter fraction ¼ is used to attenuate these fluctuations. The filter fraction is a compromise between amplitude attenuation and system response time.

In a first "learning" stage, the user head rest position (center) and size (boundaries) are extracted from 100 webcam images (no object displayed). This phase essentially calibrates for the typical working position of the user for a given PC system. This step needs to be performed only once, as the rest position and size data can be stored for later use of the same setup. It was tested whether this initial calibration data can be used for another person (see results).

After these initial steps for calibration of the webcam, the 3D dataset is loaded into memory[13] and displayed on the computer screen. The mesh triangles are rendered and visualized using the standard Matlab 3D drawing command "patch". Corresponding spectral absoprtion and light reflectance properties are added to the resulting 3D surface. Control of the webcam is added to Matlab using the MATLAB Support Package for USB Webcams.

The webcam is used to continuously acquire images, extract the head position from the webcam images, and adjust the rendering of the 3D dataset corresponding to the position of the user – and thus to the view direction and proximity – in front of the screen. Performance tests are done on 200 images per test settings in order to estimate the frame rate of the system. In some instances, the face can not be properly detected due a number of reasons such as blending of the face boundaries with the background, light reflections from the face surface, etc. As the algorithm continuously tries to find the face in the image, a noticeable dead time with a drastic drop in video frame rate can be observed. This is perceived as an unacceptable slow response of the viewer application. In order to test and further evaluate this behavior, we included for each of the settings, a test phase, during which faces are effectively hidden from detection by shielding parts of the face[14,15].

4. RESULTS

The results of the performance tests for the large 3D dataset using a mesh with 750 000 vertices are shown in Table 1, which summarizes the computation times for four different settings of the algorithm, i.e. two different webcam image resolution settings (small: 160x120pix, medium: 640x480pix) and two different settings of the Viola Jones algorithms (unconstrained face search or specification of a minimum face size). In the columns, the computation times for the individual parts of the algorithm (acquisition and haar feature detection, extraction of the user position and proximity, adapted rendering of the 3D dataset) are given. The overall computation time is typically below 40 ms for one complete loop of the algorithm including webcam data acquisition, image processing, and 3D rendering. This corresponds to a video rate of ca. 25 fps, which is fast enough for realistic rendering and smooth transitions. As a compromise between image acquisition speed and contrast-to-noise ratio[16], the webcam is operated at a fixed exposure time of 2^{-9} s.

Table 1. Computation and visualization time for a large 3D dataset (750 000 vertices) for different settings

settings	acquisition and detection (ms)		extraction (ms)	rendering (ms)	
	face detection	face not detected	position	translation	zoom
160x120 pix unconstrained	31.9±7.5	33.2±7.7	0.05±0.01	2.5±1.0	0.3±0.1
160 x 120 min face size	30.0±7.7	34.0±9.8	0.06±0.05	3.0±1.5	0.3±0.2
640 x 480 unconstrained	360±51	345±33	0.1±0.1	3.4±2.0	0.3±0.2
640 x 480 min face size	39.4±6.9	33.6±10.0	0.08±0.08	6.2±2.7	0.5±0.4

Similar to the results above, Table 2 shows a summary of the viewer application performance when displaying a rather compact 3D dataset (with mesh size of 1 500 vertices). These results indicate that the effect on the overall performance does not largely depend on the size of the rendered dataset.

Table 2. Computation and visualization time for a rather compact 3D dataset (1 500 vertices) for different settings

settings	acquisition and detection (ms)		extraction (ms)	rendering (ms)	
	face detection	face not detected	position	translation	zoom
160x120 pix unconstrained	31.5±6.8	33.1±7.8	0.06±0.03	3.8±2.2	0.4±0.3
160 x 120 min face size	31.1±7.9	34.1±8.5	0.05±0.01	2.7±1.2	0.3±0.1
640 x 480 unconstrained	360±33	341±38	0.52±0.15	3.3±1.6	0.3±0.2
640 x 480 min face size	32.2±5.7	33.4±10.0	0.1±0.1	5.0±1.6	0.5±0.4

Further tests include the use of calibration data (initial measurement of rest position and size of the user head) for use of the viewer application by another person. Users reported no significant differences when using either their own calibration set or that of a different user (data not shown). In addition, the operating distances were evaluated, i.e. the minimum and maximum distance between the user and the webcam. The minimum distance between user and webcam are given in Table 3. In our settings, the webcam was always mounted centrally on top of the computer monitor.

Table 3. Operating distances between user and webcam

settings	distance (cm)	
webcam	minimum	maximum
Microsoft LifeCam VX-2000	22	95
Microsoft LifeCam Studio	45	115
Integrated Webcam	33	90

The minimum operating distance of approximately 22 cm was obtained when using the Microsoft LifeCam Studio. If the operator moves closer, the standard face detection algorithm applied is no longer able to detect the face from the acquired image. In contrast, the maximum distance was found to be approximately 115 cm when using the Microscoft LifeCam VX-2000. All values are compatible for the values recommended for the operating distance between user and monitor, which are typically in the range between 50 and 80 cm.[17,18] This allows for comfortable use of the software, as the user does not need to change his/her routine layout of the computer hardware.

5. DISCUSSION

In this work we have proposed and implemented a robust adaptation approach for the VR rendering of 3D image data. In contrast to conventional approaches for user-position aware display of 3D datasets, which rely e.g. on stereo vision glasses, our approach does not require any additional hardware except for a simple webcam. In principle, our approach is scalable to make use of multiple camera modules and multiple monitors, so that a much larger range for the angle of view is possible. Presently, using a single webcam, the range for the tilt angle is limited to roughly ±20°. This limit can be overcome by rotating the rendered object by a multitude of the real angle of direction, such that the object is e.g. rotated by twice the angle corresponding to the displacement of the user head in the webcam image, although this type of rendering gives a much less realistic impression. In the current implementation, the adjustment of the viewing direction (for the horizontal and lateral offset) shows a greater accuracy than the adjustment of the viewing angle (zoom). The associated fluctuations in magnification were largely mitigated by the use of an infinite impulse response (IIR). However, as a consequence the response of the zoom as the user moves closer to the screen exhibits a noticeable lag time by reducing the corresponding update rate for the zoom value to about 6-7 fps (while maintaining a frame rate of 25 fps, and an update rate of 25 fps for the horizontal and vertical rotation).

A typical application of face detection is the estimation of the head pose, which consists of 3 degrees of freedom for the head position (x,y,z) and three degrees of freedom for the tilt of the head (rotation around the three axes). Additional degrees of freedom may be introduced by considering the direction of gaze. In our setting, we effectively reduced the number of degrees of freedom to match our application. We assume that the user is looking at the screen, hence we

neglected gaze direction. We are also not interested in extracting the tilt of the head, because the viewing direction towards the screen is only marginally changed upon a change in head tilt. A similar approach for face detection was also used in a student project in 2018 in a course on machine vision[19].

The viewer application presented here was tested using 3D data from various settings, e.g. from architecture (e.g. the Pantheon), from surface rendered data of embryonic bone structure, from sculpture, from space, and from common CAD models. So far, we have not tested the implementation with data from arts and performances.

We found that the viewer application presented does not provide a user experience close to a proper 3D VR/AR system, such as those presently applied in teaching or gaming. The reason for this is that the rendering of the 3D dataset on the 2D computer monitor lacks the possibility to include 3D depth information. However, for reasons laid out in the introduction, the system was designed to work without additional headgear, and can be easily implemented in different external rugged and dirt/water resistant devices. Presently, the system adjusts the rendered data for a single user. If multiple users are present in the acquired webcam images, the system finds the biggest face detected, and optimizes the display for this face.

Theoretically, a system geometry with the webcam in the back of the user is also possible. However, the algorithms for face detection are much more robust than those for body part detection (back of the head), potentially also resulting in different specifications for the computation power required. An alternative to conventional 2D face detection would be the use of a 3D camera system, instead of a classical webcam. These 3D sensors have shown remarkably versatile use[20] and are becoming integrated in more and more consumer multimedia products. However, presently standard PCs and laptops are not yet equipped with the technique.

The zoom algorithm, which is based on extraction of the head size from the images, could be implemented in a way, which allows the user to virtually move right inside the objects. This could be an important asset for rendering medical tomographic data[21–23] or for moving through buildings. Combining the head position and size algorithms with eye gaze tracking and/or with tracking of additional body parts (e.g. hands, fingers, arms) could render the approach into a more elaborate human interface device for intelligent human-machine interaction. In a second version of the viewer application, we have included position-sensitive areas which allow the user to perform actions when moving into these areas. The effect is similar to the mouse pointer hovering over a sensitive element on screen. For instance, when the user head is positioned at the far right in front of the computer screen, the CAD data is continuously rotated left as long as the user stays in this position. In a similar manner, different ranges for the parameters (head position and size) could be attributed with special tasks to be performed when the user moves in the corresponding parameter range. Such additional features could be used to e.g. tilt, pan, rotate the object if the user head is positioned over the corresponding sensitive area. Another approach to increase the angular range for head position extraction would be to add special optics to the webcam[24], in order to increase the view angle of the webcam.

In this paper, we proposed a robust, adaptive 3D rendering approach based on a quasi VR display keeping track of the user viewing direction and proximity to the displayed object. The approach could be shown to work for datasets differing by orders of magnitude in size. Unnecessary dead time due to the Viola Jones algorithm not being able to detect a person in the image, and the resulting delay of the display update, could be reduced by a large amount by either scaling down the image (i.e. using a low resolution webcam image of 160 x 120 pixels) or by specifying a minimum size of the face in the webcam image to the Viola Jones algorithm. The second approach can achieve a better accuracy when extracting the head position and size, without affecting performance of the algorithm.

Overall, users have found that the system is very intuitive and easy to use, as it requires no direct interaction of the user with the software. The software does not present any form of options or settings to the user; these values are extracted in an initial calibration step, and the calibration data of a single experiment can potentially be used for all users of the system. The approach requires no additional hardware, it can be implemented in various architectures supporting virtually all presently available devices from all manufacturers. A demo version based on Matlab code is available for download[25]. Experiments show that the algorithm provides real-time user-position aware rendering of large 3D datasets at a frame rate of ca. 25 fps.

REFERENCES

[1] Mann, S., Furness, T., Yuan, Y., Iorio, J. and Wang, Z., "All Reality: Virtual, Augmented, Mixed (X), Mediated (X,Y), and Multimediated Reality" (2018).

[2] Johnny Chung Lee., "Johnny Chung Lee - Projects - Wii," 2008, <http://johnnylee.net/projects/wii/> (2 July 2018).

[3] Lantz, E., "The future of virtual reality: head mounted displays versus spatially immersive displays (panel)," Proc. 23rd Annu. Conf. Comput. Graph. Interact. Tech., 1–2 (1996).

[4] Mon-Williams, M., Plooy, A., Burgess-Limerick, R. and Wann, J., "Gaze angle: a possible mechanism of visual stress in virtual reality headsets," Ergonomics **41**(3), 280–285 (1998).

[5] Fuchs, P., [Virtual Reality Headsets], CRC Press, CRC Press/Balkema P.O. Box 11320, 2301 EH Leiden,The Netherlands (2017).

[6] Rosenfeld, A. and Kak, A. C., [Digital picture processing], Academic Press (1982).

[7] Edwards, E. K., Rolland, J. P. and Keller, K. P., "Video see-through design for merging of real and virtual environments," Proc. IEEE Virtual Real. Annu. Int. Symp., 223–233, IEEE.

[8] Jasek, M., Pioch, N. and Zeltzer, D., "Enhanced visual displays for air traffic control collision prediction," Anal. Des. Eval. Man–Machine Syst. 1995, 553–558 (1995).

[9] Bolas, M. T., "Human factors in the design of an immersive display," IEEE Comput. Graph. Appl. **14**(1), 55–59 (1994).

[10] Viola, P., Viola, P. and Jones, M., "Robust Real-time Object Detection," Int. J. Comput. Vis. (2001).

[11] "NASA 3D Resources.", <https://nasa3d.arc.nasa.gov/models/printable> (1 June 2018).

[12] "OpenCV library.", OpenCV 3.3.1, 2017, <https://opencv.org/> (2 June 2018).

[13] Riley, D., "CAD2MATDEMO.M - File Exchange - MATLAB Central," <https://ch.mathworks.com/matlabcentral/fileexchange/3642-cad2matdemo-m?focused=5053288&tab=function> (4 June 2018).

[14] Chriskos, P., Munro, J., Mygdalis, V. and Pitas, I., "Face Detection Hindering," Signal Inf. Process. (GlobalSIP), 2017, IEEE (2018).

[15] Sharif, M., Bhagavatula, S., Bauer, L. and Reiter, M. K., "Accessorize to a Crime," Proc. 2016 ACM SIGSAC Conf. Comput. Commun. Secur. - CCS'16, 1528–1540, ACM Press, New York, New York, USA (2016).

[16] Birk, U. J., [Super-resolution microscopy : a practical guide, 1st ed.], Wiley VCH, Weinheim (2017).

[17] Unfallversicherungsanstalt, S., "Die Arbeit am Bildschirm" (2003).

[18] VBG, V.-B., "Büroarbeit – sicher, gesund und erfolgreich," Hamburg (2016).

[19] Birk, U., Andres, M., Kessler, V., Linvers, C., Zbinden, D., Catregn, G.-P. and Leutenegger, T., "Image Processing and Machine Vision for Engineering Undergraduate Students," ICMV 2018, V024–A, Munich (2018).

[20] Merbold, H., Catregn, G.-P. and Leutenegger, T., "Time-of-flight range imaging for underwater applications," Photonic Instrum. Eng. V, Y. G. Soskind, Ed., 2, SPIE (2018).

[21] Birk, U. J., Darrell, A., Konstantinides, N., Sarasa-Renedo, A. and Ripoll, J., "Improved reconstructions and generalized filtered back projection for optical projection tomography," Appl. Opt. **50**(4) (2011).

[22] Rieckher, M., Birk, U. J., Meyer, H., Ripoll, J. and Tavernarakis, N., "Microscopic optical projection tomography in vivo," PLoS One **6**(4) (2011).

[23] Sarasa-Renedo, A., Favicchio, R., Birk, U., Zacharakis, G., Mamalaki, C. and Ripoll, J., "Source intensity profile in noncontact optical tomography," Opt. Lett. **35**(1) (2010).

[24] Sigel, A., Maillard, P., Rank, M. and Heinrich, A., “Individualisierte optische Messtechnik basierend auf additiv gefertigten optischen Komponenten,” tm - Tech. Mess. **84**(7–8), 512–524 (2017).

[25] Birk, U. and Roebrock, P., “Gitlab: Matlab implementation of user-position aware display of a 3D dataset,” adaptViewTiltFast2.m, 2018, <https://gitlab.com/htw-chur/adaptive_3D_rendering> (5 June 2018).