# A Semi-Automatic Markerless Augmented Reality Approach for On-Patient Volumetric Medical Data Visualization

Márcio C. F. Macedo and
Antônio L. Apolinário Jr.
Federal University of Bahia - UFBA
Salvador, Bahia, Brazil
Email: marciocfmacedo@gmail.com,
apolinario@dcc.ufba.br

Antonio C. S. Souza
Federal Institute of Bahia - IFBA
Salvador, Bahia, Brazil
Email: antoniocarlos@ifba.edu.br

Gilson A. Giraldi
Nat. Lab. of Scientific Computing - LNCC
Petrópolis, Rio de Janeiro, Brazil
Email: gilson@lncc.br

***Abstract*—In general, augmented reality (AR) applications use polygonal models to augment a real scene. However, most of the medical applications have volumetric data to be visualized. Therefore, it is desirable to the medical AR systems to provide the visualization of these volumes into the real scene in real-time.**

**In this paper we introduce a real-time semi-automatic approach for on-patient medical volume data visualization. It is proposed in a markerless augmented reality (MAR) environment and the medical data consists of a volume reconstructed from 3D computed tomography (CT) image data. A 3D reference model of the region of interest in the patient is generated and tracked from the Kinect depth stream. From the estimated camera pose, a volumetric medical data can be displayed to a physician inside the patient's anatomy at the location of the real anatomy.**

**We evaluate the use of the named standard volume rendering techniques in the context of a MAR environment. The results obtained so far show that these techniques can be applied in this scenario in real-time.**

***Index Terms*—Augmented Reality; Volume Rendering; Markerless Registration;**

## I. Introduction

Augmented Reality (AR) is a technology in which the view of a real scene is augmented with additional virtual information. Accurate tracking, or camera pose estimation, is not only one of the fundamental requirements for the registration of virtual objects, but also is one of the most important technical challenges of AR systems.

Typically, patient's anatomical structures are shown on monitors and based on 2D images, corresponding to slices of the 3D data. In this case, the physician has to mentally compose what is shown on the screen to the patient. AR takes over this task of mental mapping by transferring it to a computer system. Therefore, the physician is able to visualize, at the same time, the patient and a part of his own anatomy in the display. This issue is addressed by some medical augmented reality systems, such as those presented in [1], [2], [3] and [4]. In general, these systems aim to improve medical diagnosis, surgical operation and/or post-operative examination.

AR applications can be divided into two basic groups: marker-based or markerless. Marker-based AR uses fiducial markers as a point of reference in the field of view to help the system to estimate the camera pose. These markers need to exist in the real world to be tracked by the application. Markerless AR (MAR) uses a part of the real scene as the marker. Tracking becomes more complex in MAR. However, because artificial markers are not part of the original scene, MAR is desirable in several AR application scenarios.

Most of the medical AR systems use bulky equipment such as optical tracking systems based on markers to help in the tracking of the 3D medical data. While they give accuracy in the tracking and positioning of the virtual medical data in the patient, in some applications these markers can be intrusive and the hardware too expensive. To solve these issues, markerless tracking can be implemented taking advantage from the visual features of the scene, such as texture or geometry of the object of interest to be tracked. Currently, the high computational power of hardware, mostly based on parallel computing, like GPUs, allied to low-cost capture devices, allows this solution to be used with enough accuracy and in real-time.

In general, AR applications use polygonal models to augment a real scene. However, medical data is typically represented by volumetric models. Therefore, it is desirable to a medical AR system to provide the visualization of these volumetric data into the AR environment in real-time.

This paper presents a semi-automatic markerless augmented reality approach for on-patient medical volume data visualization based on a low-cost depth sensor available in the Kinect device [5]. First, a 3D reference model of the region of interest in the patient is generated. Next, the user positions the virtual object (i.e. medical data) into the reference model. Afterwards, the Kinect raw data is aligned to the 3D reference model, predicting the current camera pose. From the estimated camera pose, a volumetric medical data can be displayed to a physician at the location of the real anatomy. The volume visualization is performed by the standard volume rendering techniques [27]. Finally, to improve the visual perception and interaction between the real (i.e. patient's image) and virtual objects,

our approach supports occlusion handling and blending at the shader level by using depth and color buffers. The final result is shown in the physician's monitor.

Our approach is evaluated in a scenario where the patient's head is augmented with a CT volumetric dataset of a head. The Viola-Jones face detector [6] is used to locate the patient's face in the whole image. The KinectFusion [7] algorithm is used to reconstruct the 3D reference model of the patient's face and the Iterative Closest Point (ICP) is used to track it. Taking advantage from this scenario, we further improve the robustness of the tracking step by using the face tracking solution proposed in [8]. Our main contribution is the evaluation of the applicability of standard volume rendering techniques in a MAR environment in real-time.

The rest of the paper is arranged as follows. Section 2 provides a review on the related work of Medical Augmented Reality systems. Section 3 presents the proposed approach. Section 4 introduces the volume rendering techniques used in this work. Section 5 discusses the experimental results. The paper is concluded in Section 6, with a summary and discussion of future work.

## II. Related Work

Medical augmented reality systems for on-patient visualization have been driven by different approaches in recent years.

Kutter et al. [9] proposed a marker-based method for real-time high quality on-patient visualization of volumetric medical data on a Head Mounted Display (HMD). Their work focuses on efficient implementations for high quality volume rendering in an augmented reality environment. They also provide occlusion handling for physician hands. An improved version of this work was proposed by Wieczorek et al. [10] to handle with occlusions due to medical instruments as well. Also, they included additional effects in the system, such as virtual mirror and multi-planar reformations.

Debarba et al. [11] proposed a method to visualize anatomic hepatectomy (i.e. anatomic liver resections) in an AR environment. The use of a fiducial marker made possible the positioning and tracking of the medical data in the scene. A mobile device was used to allow the visualization of internal structures of the patient's body.

Lee et al. [12] proposed a markerless registration framework for a medical augmented reality system. They use three cameras: two of them are mounted to form a stereo vision system and reconstruct the patient's head; the other camera is used to capture the images of the patient in real-time. In a preprocessing step, a surface is reconstructed from the CT and a variant of the ICP algorithm is used to do the image-to-patient registration. The estimation of the third camera's pose is done by using a fiducial marker.

Suenaga et al. [13] proposed a method for on-patient visualization of maxillofacial regions. A 3D optical tracking system and a fiducial marker are used to track the patient. A semi-transparent display is placed in front of the mouth region of the patient. The display shows the maxillofacial medical data. This method runs in 5 FPS (frames per second).

Different from the previous approaches, Maier-Hein et al. [4] proposed a method for markerless mobile augmented reality for on-patient visualization of medical images. They proposed a system in which a Time-of-Flight (ToF) camera is mounted on a mobile and portable device (e.g. tablet PC, iPad) and the physician might move the portable device along the body of the patient to see his anatomical information. To estimate the camera pose, they use a graph matching procedure [14] and a anisotropic variant of the ICP algorithm [15] to align the surfaces continuosly captured by the ToF camera. This method runs in 10 FPS.

In the field of anatomy education, Blum et al. [16] proposed the *mirracle*, a magic mirror for teaching anatomy. They used a display device and a Kinect sensor to allow volume visualization of a CT dataset augmented onto the user. To track the pose of the user, they used the NITE skeleton tracking [17]. As the system is for educational purposes, they could use a generic CT volume which was scaled to the size of the user and augmented onto him.

Most of the approaches described in this section share the same drawback: they use markers to help in the calibration, positioning and tracking of the objects in the scene. The use of fiducial markers provides fast and accurate tracking. However, these markers are still intrusive, because they are not part of the original scene. Moreover, the hardware of the optical tracking system in some applications is too expensive. The approach that does not use these marker-based hardware (i.e. [4]), does not obtain real-time performance in its application because of the computational cost of the markerless tracking in conjunction with the volume rendering techniques used. One exception of this is the *mirracle* system described before. However, the main drawback of the fast NITE skeleton tracking is that it does not track accurately some parts of the body, such as head. Different from them, our approach based on a markerless tracking runs entirely in real-time with low-cost hardware components. Moreover, the solution proposed in this paper is general in the sense that can track any part of the body with enough accuracy.

## III. On-Patient Volumetric Medical Data Visualization

In this section we describe the MAR environment in which this work was based on. An overview of this environment with support to volumetric data can be seen in Figure 1.

### A. Environment setup

The proposed approach is based on an RGB-D sensor and a computer with GPU, therefore being accessible to any user. The Kinect [5] is used as RGB-D sensor to capture the patient's color and depth information. The real-time performance is achieved by using the parallel processing power of the GPU in all critical algorithms.

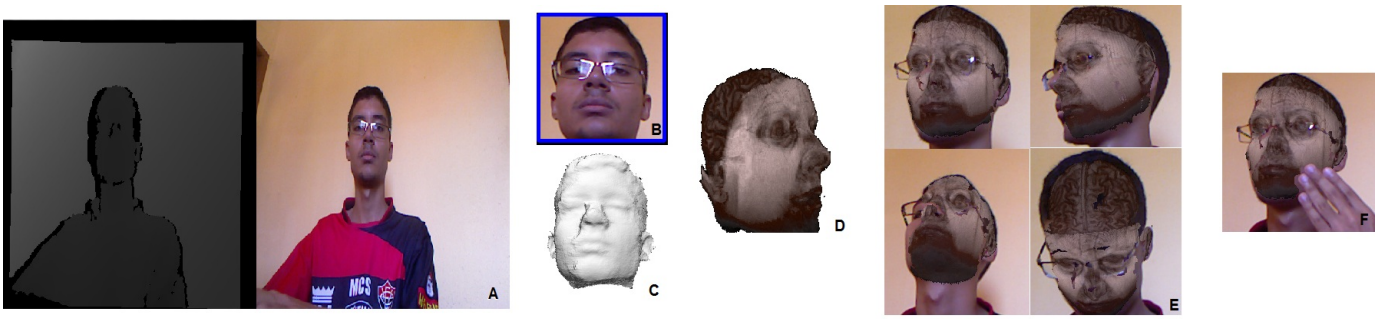


Fig. 1. Overview of the processing pipeline. A) RGB-D live stream. B) A face detector is used to locate and segment the face from the rest of the scene. C) 3D Reference model is reconstructed with KinectFusion. D) The 3D reconstruction is stopped, the volume is rendered and the user positions the virtual object into the scene. E) MAR tracking is done based on the 3D reference model. F) Our approach supports occlusion.

### *B. Vertex and Normal Maps Generation*

In this section we describe the algorithms that are used for every input frame to generate the vertex map[1] and the normal map[2], with the exception of the face segmentation, which is only performed during the reconstruction stage. All of these algorithms run on the GPU.

As mentioned before, the Kinect has two sensors that capture color and depth information of the scene (Figure 1-A). Therefore, we calibrate the Kinect sensors to enable a mapping between them.

In order to segment the face from the scene, we apply the Viola-Jones face detector [6] implemented in GPU to locate and segment the face in the color image (Figure 1-B). As long as the face is segmented, its location can be transposed to the depth image by using the extrinsic parameters from the calibration step. Doing so, we achieve a more restricted area of the depth map to be used through the other steps of our approach.

The depth map is denoised using a bilateral filter [18] that preserves discontinuities of the raw depth map. We segment the depth of the background scene by applying a Z-axis threshold on the filtered depth map. This threshold is defined semi-automatically.

The filtered depth map is then converted into a vertex map and a normal map. The vertex map is constructed by the product between the filtered depth map and the Kinect IR camera's intrinsic calibration matrix. The normal map is constructed by computing the eigenvector of smallest eigenvalue of the local covariance matrix computed for every vertex. This technique produces normal maps with less error than the traditional approaches based on neighboring points for normal estimation [19].

### *C. 3D Reference Model Reconstruction*

With the real object properly segmented, we need to track it through the Kinect live stream (whose algorithm is described in the Section III-D) and to integrate the different viewpoints acquired from the scene into a single reference model.

[1]Point cloud.

[2]In this paper, we refer to normal map as an array which stores the normal vector for each vertex in the vertex map.

The KinectFusion algorithm [7] is used in this context to reconstruct a 3D reference model in real-time (Figure 1-C). The algorithm integrates raw depth data from a Kinect into a volumetric grid to produce a high-quality 3D reconstruction of a scene. The grid stores at each voxel the distance to the closest surface (i.e. Signed Distance Function - SDF) [20] and a weight that indicates uncertainty of the surface measurement. The SDF values are positive in-front of the surface, negative behind and zero-crossing where the sign changes. In the KinectFusion, the SDF is only stored in a narrow region around the surface, in other words, a truncated SDF (TSDF). These volumetric representation and integration are based on the VRIP algorithm [21]. Surface extraction is achieved by detecting zero-crossings through a ray caster. All these operations run on the GPU. As shown in [22], the KinectFusion algorithm has accuracy of $10mm$ and so on we assume that its reconstructed models are adequate to be used as reference for registration in AR applications.

The 3D reference model reconstruction is done only one time and it is the basis for the MAR live tracking. The reconstruction is stopped semi-automatically and the user can position the medical data (Figure 1-D) into the scene. Afterwards, the MAR tracking can be started.

### *D. Live Tracking*

The live tracking is done in two steps: during the reconstruction of the 3D reference model and during the MAR with the patient and the medical data (Figure 1-E). Instead of using an image-based tracking, we use a real-time variant of the Iterative Closest Point (ICP) [23], [24]. As done in [25], we choose a depth-based method because it does not suffer from changes in illumination or the presence of textureless regions in the scene. The ICP is used to estimate the transformation that aligns the current depth frame with the previous one represented by the 3D reference model.

In presence of fast rotations and translations in the scene, the real-time variant of the ICP algorithm may fail. To minimize this problem, a real-time head pose estimation algorithm is used to give a new initial guess to the ICP to compute correctly the current transformation [8]. The head pose estimation used is the algorithm proposed by Fanelli et al. [26]. This

probabilistic approach achieves high accuracy and runs in real-time even running in CPU.

After the computation of the rigid transformation, it is applied to the medical data, which can be composed with the real scene captured by the Kinect.

## IV. Volume Rendering

Volume rendering is concerned with techniques for generating images from volume data [27].

The majority of volume rendering algorithms are based on the volume rendering integral. This formulation is based on a emission-absorption optical model as shown in Equation 1.

$$I(D) = I_0 e^{-\int_{s0}^{D} k(t)dt} + \int_{s0}^{D} q(s) e^{-\int_{s}^{D} k(t)dt} ds. \quad (1)$$

The radiance energy *I(D)* is the result of integrating from entry point into the volume ($s = s_0$) to the exit point toward the camera ($s = D$). The absorbed energy and emission components are represented by the absorption and emission coefficients $k$ and $q$ respectively. The term $I_0$ is the radiance in the entry point $s_0$.

The volume is rendered according to a compositing scheme, which gives the numerical computation of the volume rendering integral:

$$I(D) = \sum_{i=0}^{n} c_i \prod_{j=i+1}^{n} T_j \quad (2)$$

where $c_i = I(s_i)$ and $T_i = T(s_{i-1}, s_i) = e^{-\int_{s_{i-1}}^{s_i} k(t)dt}$.

Two of the most known compositing schemes are the direct volume rendering (DVR) and the maximum intensity projection (MIP). The DVR is the discretization presented in the Equation 2 and it is based on a front-to-back or back-to-front compositing. The most common is the front-to-back DVR:

$$C_{dst} = C_{dst} + (1 - \sigma_{dst}) C_{src} \quad (3)$$

$$\sigma_{dst} = \sigma_{dst} + (1 - \sigma_{dst}) \sigma_{src} \quad (4)$$

where $C_{dst} = c_{i+1}$, $C_{src} = c_i$, $\sigma_{dst} = 1 - T_{i+1}$, $\sigma_{src} = \sigma_i$ given the voxel $i$ being traversed. $C$ represents the color contribution and $\sigma$ the opacity of the voxel.

Different from the DVR compositing scheme, MIP is computed according to the following compositing equation:

$$C_{dst} = max(C_{dst}, C_{src}) \quad (5)$$

The final result is the maximum color contribution along a ray [27]. This compositing scheme is particularly important in the virtual angiography (i.e. the display of the vessel structures in medical scans) [28].

The volume data is represented as a 3D texture with associated colors. This representation allows the generation of images with higher quality than the 2D texture-based solution [27]. To render the medical data based on DVR or MIP compositing scheme, the ray casting technique is used. The start positions of the ray are obtained by rasterizing the front faces of the volume bounding box and the exit positions of the ray are obtained by rasterizing the back faces of the bounding box. Direction is computed from the difference between the exit and start positions. Ray casting is performed by sampling the space in-between the volume bounding box by using an adaptive sampling rate (which is discussed below). Ray casting is done on GPU in a single rendering pass on the fragment shader [29].

One of the main advantages of the ray casting is that it is flexible in the sense that many other techniques can be integrated to improve the image quality or the performance of the rendering.

To reduce the sampling artifacts a stochastic jittering (i.e. random ray-start off-setting) is applied to the ray start position. To reduce the filtering artifacts a fast GPU-Based tri-cubic filtering [30], [31] and a GPU pre-filter for accurate tri-cubic filtering [32] are used.

The performance of our volume rendering is optimized by empty-space leaping the non-visible voxels. The volume is subdivided into an octree. In order to detect empty space, each block stores the minimum and maximum scalar values. The visibility of each block can be determined after evaluation of the transfer function [33]. If the block is considered invisible, the step size of the ray is increased, otherwise, it is decreased. Our approach also supports early ray termination, if the opacity accumulated is greater than a threshold, and image downscaling, when the volume size is not supported by the graphics rendering.

The volume data consists in scalar values that represent some spatially varying physical property. Transfer functions can be applied on these scalar values to improve the user's visual perception and data interpretation of the volume. The transfer functions map the values to colors in the RGB space. In this work, pre-integrated transfer functions [34] are used to capture the high frequencies introduced in the transfer functions with low sampling rates.

The volume rendering integral presented in the Equation 1 does not account for illumination effects caused by external light sources. Such illumination effects, however, add a great deal of realism to the resulting images. This is specially important in an AR environment, where this illumination effect serves as an approximation of the illumination of the real scene. To compute the local illumination, it is used Blinn-Phong shading [35] with on the fly gradient computation by central or forward differences on the GPU. Non-polygonal iso-surface rendering is realized by first hit ray-casting. The local illumination is included in the Equation 1 by extending the emission coefficient $q(s) = q_{ea}(s) + q_{il}(s)$, where $q_{ea}(s)$ is the emission coefficient of the emission-absorption model and $q_{il}(s)$ is the coefficient that adds the local illumination [27].

### A. Integration into a MAR environment

After the volume rendering, we read the color frame buffer of the volume and send it to a shader to blend it with the RGB


Fig. 2. Some of the visualization options. A) Direct volume rendering (DVR). B) DVR with pre-integrated transfer function. C) DVR with pre-integrated transfer function and Blinn-Phong illumination. D) Non polygonal iso surface volume rendering.

data coming from the Kinect sensor. The blending is done by the following linear interpolation:

$$I_{final} = \beta * I_{real} + (1 - \beta) * I_{medical}, \quad (6)$$

where $I_{real}$ is the image captured by the sensor, $I_{medical}$ is the image corresponding to the medical volume, and $I_{final}$ is the resulting image. In our approach, $\beta$ is 0.2.

Incorrect occlusion of virtual and real objects in an augmented scene is one of the fundamental problems in AR. To solve it we use the depth maps of the 3D reference object reconstructed previously (reference) and the 3D object coming from the sensor's live stream (live). If the live object is in front of the reference object, the volume is the occludee, otherwise, it is the occluder (Figure 1-F).

Blending and occlusion are computed in a GLSL (OpenGL Shading Language) fragment shader that process the color and depth buffers, respectively, to do these operations.

## V. Results and Discussion

In this section we analyze the performance of the whole approach and the techniques employed at the volume rendering.

For all tests we used an Intel(R) Core(TM) i7-3770K CPU @3.50GHz 8GB RAM and a NVIDIA GeForce GTX 660. We used the open source C++ implementation of the KinectFusion [36] released by the PCL project [37].

The medical dataset used is a CT volumetric data of a head [38] of two different resolutions: $256^3$ and $512^3$. The reference human face was reconstructed with the KinectFusion using a grid with volume size of $70cm$x$70cm$x$140cm$ and resolutions of $256^3$ and $512^3$.

We evaluate the performance of our approach in a scenario where the patient's head is augmented with a CT volumetric dataset of a head. The use of a generic volume does not affect our evaluation, as our main interest is to evaluate the performance of the proposed approach in the context of a medical MAR environment.

Figure 2 shows the volume rendered with different visualization options. Figure 3 shows the influence of some techniques to improve the image quality of the volume rendering.

In the first test, the time required for each step of our approach was measured in a situation with the resolution of the KinectFusion's grid in $512^3$ and of the medical data in $256^3$. The goal of this test is to evaluate the performance for each step individually.

Fig. 3. A volume rendering (left) with stochastic jittering (center) and tri-cubic filtering (right). The stochastic jittering reduces the wood-grain artifacts in the volume, however it is almost imperceptible in this scene. The tri-cubic filtering smoothes the volume data, reducing the artifacts present in the volume rendered with trilinear filtering.

In our preprocessing computation, the 3D reference model reconstruction takes 23 ms per frame (43 FPS) and requires less than 15 seconds to be completed. In the MAR live tracking, the time measured for each step of our approach can be seen in Figure 4. From there, we observe that the application takes 21 ms per frame (47 FPS). As the Kinect provides depth maps at 30 FPS, our approach can process every input frame coming from the RGB-D sensor. Therefore, we can conclude that it runs in real-time.

The vertex and normal map generation takes more time compared to the original results obtained from the KinectFusion implementation [39]. It occurs because our normal map computation is based on the covariance matrix, which is a method more expensive than the traditional construction by the cross product between neighboring vertices used in the original KinectFusion. As mentioned before, this normal map computation produces more accurate normal maps [19].

The computation of the occlusion, which transfers the data stored in the GPU to the CPU, converts the 3D reference object and 3D object coming from the Kinect to the same coordinate system and sends their depth maps to the shader, takes the highest time in our approach. Meanwhile, our direct volume rendering takes the lowest time.

The solution proposed in [8] was not included in the measurement of the live tracking, because it does not occur for every input frame. When used, it added 40 ms in the total frame time. Moreover, we have observed that the user takes less than 10 seconds to position and adjust the volume in the scene.

In the second test, we evaluate the influence of the volume

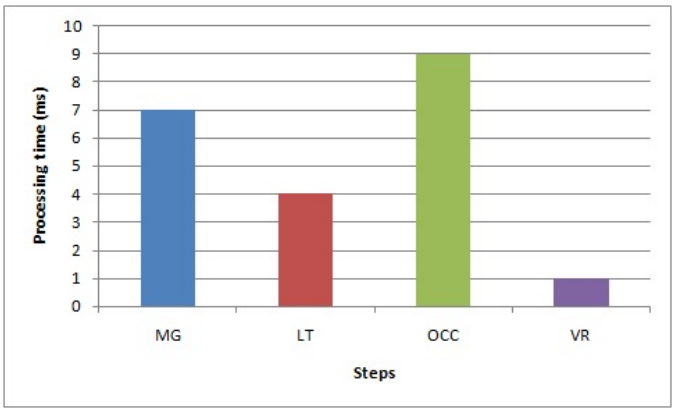


Fig. 4. Performance results measured in average milliseconds for each step of our approach. MG - Vertex and Normal Map Generation. LT - Live Tracking. OCC - Occlusion. VR - Volume Rendering. Times were measured running our approach with the KinectFusion's grid in resolution $512^3$ and the medical data in $256^3$.

dimensions on the overall performance of our approach. Our approach never dropped below 29 FPS using the medical and KinectFusion volumes of resolution $256^3$ or $512^3$ with DVR. Therefore, we can use the maximum KinectFusion's volume size to generate a more accurate 3D reference model.

In the third test, the average processing time for various volume rendering modes was measured. The performance results can be seen in Figure 5. From the analysis on the first test, if the volume rendering takes less than 10 ms the application still runs in real-time. Considering that the typical resolution of a head medical volume is $256^3$ [40], we can conclude that our approach runs in 30 FPS once the volume rendering mode which takes most time in this resolution needs only 7 ms. However, with a volume of resolution $512^3$, depending on which mode is chosen, we have a loss in the performance of the application.

As described in Section IV, the volume is stored as a discrete 3D texture. When the ray is casted into the volume, it accesses the space between the discrete samples of the volumetric data. In this case, the trilinear interpolation is used to reconstruct a continuous representation of the volume based on the eight closest neighbours samples of that space. This is the most expensive operation in the volume rendering based on ray casting as it requires eight memory access to perform the interpolation. Based on this statement, it is possible to evaluate the variation in performance of the different volume rendering modes.

In the simplest DVR, the trilinear interpolation is performed only once for each position of the ray casted. Therefore, this is the rendering mode which takes the lowest processing time. As consequence, it produces the simplest visual effects, which can be seen in Figure 2-A.

In the non-polygonal iso surface rendering, the Blinn-Phong shading is computed when the ray traverses a voxel with isovalue greater than a threshold defined semi-automatically. The normal vector for a given voxel is computed by the normalization of the central differences of the neighbouring voxels. This gradient estimation requires six trilinear interpolations. However, as it is not computed for every voxel being traversed, it does not increase significantly the computational cost of the volume rendering. An example of non-polygonal iso surface rendering can be seen in Figure 2-D.

In the DVR with pre-integrated transfer function, after the trilinear interpolation of the voxel, the scalar value of the previous and current voxel being traversed are used as a look-up in a 2-D pre-integration table. This lookup is performed with a bilinear interpolation. It increases the volume rendering processing time to 15 ms per frame and decreases slightly the performance of the application to 35 ms per frame (28 FPS). As a result, we have a more pleasant visual of the volume. An example of such effect can be seen in Figure 2-B.

In the DVR with transfer function and local illumination, for every voxel being traversed, the transfer function is accessed (with a bilinear interpolation) and the illumination is computed (with six trilinear interpolations). These interpolations decrease significantly the performance of the volume rendering to 30 ms per frame and the application to 50 ms per frame (20 FPS), which is not prohibitive, as the user can still interact with the application with some delay. In the final result, the illumination effects add realism to the resulting image. An example can be seen in Figure 2-C.

In the test performed with a simple DVR and fast tri-cubic filtering, for every voxel being traversed, eight trilinear interpolations are computed to return one tricubic interpolation. These interpolations decrease the performance similarly to the situation of transfer function and local illumination. The volume requires 30 ms per frame to be rendered and the application requires 50 ms per frame (20 FPS). The influence of the tri-cubic interpolation instead of the trilinear one can be seen in Figure 3.

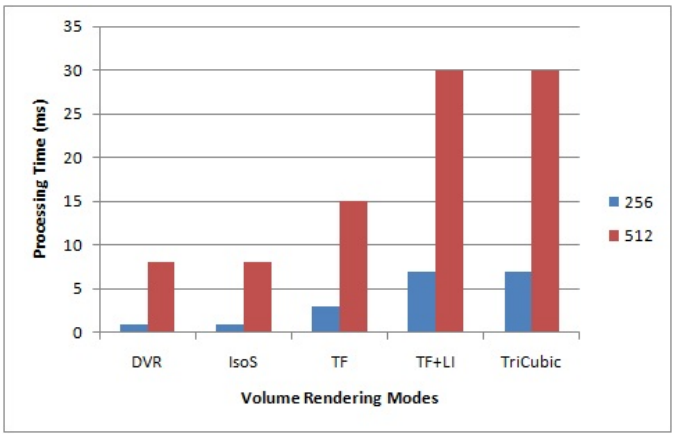


Fig. 5. Performance results measured in average milliseconds for various volume rendering modes. DVR - Direct Volume Rendering. IsoS - Non-Polygonal Iso Surface. TF - DVR + Transfer Function with Pre-Integration. LI - DVR + Local Illumination via Blinn-Phong. TriCubic - Fast Tricubic filtering. 256 - $256^3$ volume. 512 - $512^3$ volume.

## VI. Conclusions and Future Work

We have presented a marker-free augmented reality approach for on-patient volumetric medical data visualization. We used the KinectFusion algorithm to reconstruct the patient's head and a variant of the ICP algorithm in conjunction with a face tracking solution to track it during the MAR. We have tested and applied standard volume rendering techniques to render volumes with good quality and shown that, with a typical volume size, the proposed algorithm is capable to run in real-time. In addition, our approach supports occlusion.

One of the current limitations of the proposed approach is that it was not evaluated with a real patient, which would enable us to validate our approach in terms of accuracy. Thus, one of the next steps must be adapt the approach to be used and evaluated in a surgical environment. Encouraged by the field of illustrative volume rendering techniques, for future work we intend to integrate into our approach methods that separate the volume into focus and context regions in order to improve the human perception of the scene. Another possibility of improvement is on the visualization of the medical volume onto the patient. One could use saliency maps or video ghosting techniques to define dynamically the contribution of each image into the final one. Also, fast global illumination could be applied to improve the realism of the volume rendering and the integration with the real scene.

## Acknowledgments

We are grateful to the PCL project for providing the open-source implementation of the KinectFusion algorithm. This research is financially supported by FAPESB and CAPES.

## References

[1] M. Schnaider, B. Schwald, H. Seibert, and T. Weller, "Medarpa–a medical augmented reality system for minimal-invasive interventions." *Stud Health Technol Inform*, vol. 94, 2003.

[2] C. Bichlmeier, F. Wimmer, S. M. Heining, and N. Navab, "Contextual anatomic mimesis hybrid in-situ visualization method for improving multi-sensory depth perception in medical augmented reality," in *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ser. ISMAR '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 1–10.

[3] L. T. De Paolis, M. Pulimeno, and G. Aloisio, "An augmented reality system for surgical pre-operative planning," in *Proceedings of the 1st WSEAS international conference on Biomedical electronics and biomedical informatics*, ser. BEBI'08. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2008, pp. 70–73.

[4] L. Maier-Hein, A. M. Franz, M. Fangerau, M. Schmidt, A. Seitel, S. Mersmann, T. Kilgus, A. Groch, K. Yung, T. R. dos Santos, and H.-P. Meinzer, "Towards mobile augmented reality for on-patient visualization of medical images." in *Bildverarbeitung für die Medizin*, ser. Informatik Aktuell, H. Handels, J. Ehrhardt, T. M. Deserno, H.-P. Meinzer, and T. Tolxdorff, Eds. Springer, 2011, pp. 389–393.

[5] L. Cruz, D. Lucio, and L. Velho, "Kinect and rgbd images: Challenges and applications," in *Graphics, Patterns and Images Tutorials (SIBGRAPI-T), 2012 25th SIBGRAPI Conference on*, 2012, pp. 36–49.

[6] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vision*, vol. 57, no. 2, pp. 137–154, May 2004.

[7] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ser. UIST '11. New York, NY, USA: ACM, 2011, pp. 559–568.

[8] M. Macedo, A. Apolinario, and A. C. Souza, "A Robust Real-Time Face Tracking using Head Pose Estimation for a Markerless AR System," in *Symposium on Virtual and Augmented Reality*, Cuiaba, MT, Brazil, May 2013.

[9] O. Kutter, A. Aichert, C. Bichlmeier, J. Traub, S. M. Heining, B. Ockert, E. Euler, and N. Navab, "Real-time Volume Rendering for High Quality Visualization in Augmented Reality," in *International Workshop on Augmented environments for Medical Imaging including Augmented Reality in Computer-aided Surgery (AMI-ARCS 2008)*. New York, USA: MICCAI Society, Sept. 2008.

[10] M. Wieczorek, A. Aichert, O. Kutter, C. Bichlmeier, J. Landes, S. M. Heining, E. Euler, and N. Navab, "GPU-accelerated Rendering for Medical Augmented Reality in Minimally-Invasive Procedures," in *Proceedings of BVM 2010*. Springer, Mar. 2010.

[11] H. G. Debarba, J. Grandi, A. Maciel, and D. Zanchet, "Anatomic hepatectomy planning through mobile display visualization and interaction." in *MMVR*, ser. Studies in Health Technology and Informatics, vol. 173. IOS Press, 2012, pp. 111–115.

[12] J.-D. Lee, C.-H. Huang, T.-C. Huang, H.-Y. Hsieh, and S.-T. Lee, "Medical augment reality using a markerless registration framework." *Expert Syst. Appl.*, vol. 39, no. 5, pp. 5286–5294, 2012.

[13] H. Suenaga, H. Hoang Tran, H. Liao, K. Masamune, T. Dohi, K. Hoshi, Y. Mori, and T. Takato, "Real-time in situ three-dimensional integral videography and surgical navigation using augmented reality: a pilot study," *International Journal of Oral Science*, no. 2, p. 98–102, 2013.

[14] T. R. dos Santos, A. Seitel, H.-P. Meinzer, and L. Maier-Hein, "Correspondences search for surface-based intra-operative registration," in *Proceedings of the 13th international conference on Medical image computing and computer-assisted intervention: Part II*, ser. MICCAI'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 660–667.

[15] L. Maier-Hein, M. Schmidt, A. M. Franz, T. R. dos Santos, A. Seitel, B. Jähne, J. M. Fitzpatrick, and H. P. Meinzer, "Accounting for anisotropic noise in fine registration of time-of-flight range data with high-resolution surface data," in *Proceedings of the 13th international conference on Medical image computing and computer-assisted intervention: Part I*, ser. MICCAI'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 251–258.

[16] T. Blum, V. Kleeberger, C. Bichlmeier, and N. Navab, "mirracle: Augmented reality in-situ visualization of human anatomy using a magic mirror," in *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE*, 2012, pp. 169–170.

[17] (2013) Openni. [Online]. Available: http://www.openni.com/

[18] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Computer Vision, 1998. Sixth International Conference on*, jan 1998, pp. 839 –846.

[19] S. Holzer, R. Rusu, M. Dixon, S. Gedikli, and N. Navab, "Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, Oct 2012, pp. 2684–2689.

[20] S. J. Osher and R. P. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, 2003rd ed. Springer, Oct. 2002.

[21] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '96. New York, NY, USA: ACM, 1996, pp. 303–312.

[22] S. Meister, S. Izadi, P. Kohli, M. Hämmerle, C. Rother, and D. Kondermann, "When can we use kinectfusion for ground truth acquisition?" in *Workshop on color-depth fusion in robotics, IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[23] P. Besl and H. McKay, "A method for registration of 3-d shapes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 14, no. 2, pp. 239 –256, feb 1992.

[24] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and Vision Computing*, vol. 10, no. 3, pp. 145 – 155, 1992.

[25] M. Macedo, A. L. Apolinario, and A. C. Souza, "A Markerless Augmented Reality Approach Based on Real-Time 3D Reconstruction using Kinect," in *Workshop of Works in Progress (WIP) in SIBGRAPI 2013 (XXVI Conference on Graphics, Patterns and Images)*, S. M. Alejandro C. Frery, Ed., Arequipa, Peru, august 2013. [Online]. Available: http://www.ucsp.edu.pe/sibgrapi2013/eproceedings/

[26] G. Fanelli, T. Weise, J. Gall, and L. V. Gool, "Real time head pose estimation from consumer depth cameras," in *33rd Annual Symposium of the German Association for Pattern Recognition (DAGM'11)*, September 2011.

[27] M. Hadwiger, J. M. Kniss, C. Rezk-salama, D. Weiskopf, and K. Engel, *Real-time Volume Graphics*. Natick, MA, USA: A. K. Peters, Ltd., 2006.

[28] M. Prokop, H. O. Shion, A. Schanz, and M. Andschaefer-Prokop, "Use of maximum intensity projections in ct angiography," in *Radiographics*, vol. 17, 1997, pp. 433–451.

[29] M. Hadwiger, P. Ljung, C. R. Salama, and T. Ropinski, "Advanced illumination techniques for gpu-based volume raycasting," in *ACM SIGGRAPH 2009 Courses*, ser. SIGGRAPH '09. New York, NY, USA: ACM, 2009, pp. 2:1–2:166.

[30] C. Sigg and M. Hadwiger, "Fast third-order texture filtering," in *GPU Gems 2*, M. Pharr, Ed. Addison-Wesley, 2005, pp. 313–329.

[31] D. Ruijters, B. M. ter Haar Romeny, and P. Suetens, "Efficient gpu-based texture interpolation using uniform b-splines." *J. Graphics Tools*, vol. 13, no. 4, pp. 61–69, 2008.

[32] D. Ruijters and P. Thévenaz, "Gpu prefilter for accurate cubic b-spline interpolation." *Comput. J.*, vol. 55, no. 1, pp. 15–20, 2012.

[33] W. Li, K. Mueller, and A. Kaufman, "Empty space skipping and occlusion clipping for texture-based volume rendering," in *Visualization, 2003. VIS 2003. IEEE*, 2003, pp. 317–324.

[34] K. Engel, M. Kraus, and T. Ertl, "High-quality pre-integrated volume rendering using hardware-accelerated pixel shading," in *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, ser. HWWS '01. New York, NY, USA: ACM, 2001, pp. 9–16.

[35] J. F. Blinn, "Models of light reflection for computer synthesized pictures," in *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '77. New York, NY, USA: ACM, 1977, pp. 192–198.

[36] (2013, Mar.) Kinfu. [Online]. Available: http://svn.pointclouds.org/pcl/trunk/gpu/kinfu/

[37] R. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, may 2011, pp. 1 –4.

[38] (2014, Feb.) The stanford volume data archive. [Online]. Available: http://www-graphics.stanford.edu/data/voldata/

[39] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ser. ISMAR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 127–136.

[40] (2014, Feb.) The volume library. [Online]. Available: http://www9.informatik.uni-erlangen.de/External/vollib/