

Occlusion Handling for Medical Augmented Reality using a Volumetric Phantom Model

Jan Fischer
Visual Computing for Medicine
WSI/GRIS
University of Tübingen
Germany

fischer@gris.uni-tuebingen.de

Dirk Bartz
Visual Computing for Medicine
WSI/GRIS
University of Tübingen
Germany

bartz@gris.uni-tuebingen.de

Wolfgang Straßer
WSI/GRIS
University of Tübingen
Germany

strasser@gris.uni-tuebingen.de

ABSTRACT

The support of surgical interventions has long been in the focus of application-oriented augmented reality research. Modern methods of surgery, like minimally-invasive procedures, can benefit from the additional information visualization provided by augmented reality. The usability of medical augmented reality depends on a rendering scheme for virtual objects designed to generate easily and quickly understandable augmented views. One important factor for providing such an accessible reality augmentation is the correct handling of the occlusion of virtual objects by real scene elements. The usually large volumetric datasets used in medicine are ill-suited for use as phantom models for static occlusion handling. We present a simple and fast preprocessing pipeline for medical volume datasets which extracts their visual hull volume. The resulting, significantly simplified visual hull iso-surface is used for real-time static occlusion handling in our AR system, which is based on off-the-shelf medical equipment.

Categories and Subject Descriptors

H.5.1 [Information Interfaces and Presentation]: Artificial, augmented, and virtual realities; J.3 [Life and Medical Sciences]: Medical information systems

General Terms

Algorithms, Human Factors

Keywords

augmented reality, occlusion handling, volume data, visual hull

1. INTRODUCTION

In augmented reality (AR), virtual graphical elements are added to the user's real surroundings. There are different

approaches to designing an AR system, which are usually divided into optical see-through and video see-through systems. While the former utilize special translucent display devices, the latter achieve reality augmentation by processing a digital video stream [1].

The planning, preparation and support of medical interventions has always been among the most important applications of augmented reality. In many modern, minimally-invasive surgical procedures, the surgeon's view of the patient anatomy and risk structures can be very limited. Dedicated medical equipment for facilitating surgical interventions with supplemental graphical representations is available as so-called image guided surgery (IGS) devices. Such devices are offered as commercial models by several manufacturers.

In addition to established standard systems for medical visualization, a number of experimental augmented reality prototypes for the support of medical treatment have been designed in recent years. Although many of these systems have reached a high level of visual quality, they usually rely on specific hardware. Such base technologies often are not well suited for use in an actual surgical intervention, since they are not certified for the clinical practice.

It is essential for any AR application to provide an augmented display that is easy to interpret by the user. A significant visual cue for understanding the spatial relationships in a scene is the correct occlusion of objects from the user's viewpoint. This is particularly important for the case of virtual objects being occluded by real objects, which is difficult to detect. If the geometry and placement of static objects in the real surroundings are known, the use of so-called phantom models is a common approach for handling the occlusion of virtual objects.

Several research groups have worked on the detection and handling of occlusion in an augmented reality rendering pipeline. Breen et al. have suggested the phantom model method for handling static occlusion in AR [2]. The AR system of Malik et al. [6] detects the occlusion of black and white marker regions by the user's hand. A method for detecting dynamic occlusion in front of static backgrounds is described in [3]. This algorithm does not require any previous knowledge about the occluding objects, but relies on a textured graphical model of planar elements in the scene. Approaches for detecting occlusion in a stereo camera AR system have also been investigated, some of which solve the occlusion problem using depth information delivered by stereo matching [7]. The method developed by Gordon et

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

VRST'04, November 10-12, 2004, Hong Kong.

Copyright 2004 ACM 1-58113-907-1/04/0011 ...\$5.00.

al. [4] can correctly render interaction devices into the scene. Correctly handling occlusion in non-real-time augmented reality was examined by Lepetit and Berger [5].

2. STATIC OCCLUSION HANDLING

In this paper, we present an approach for correctly handling the occlusion of virtual graphical objects by the patient. By solving this problem, we enable the user to easily determine whether a graphical object - like the rendering of a surgical tool or an instrument trajectory - is supposed to be in front of or behind the patient. A volumetric dataset containing the relevant part of the patient’s anatomy is acquired before the actual intervention using a medical scanning procedure. After the initial registration, the image guided surgery (IGS) system requires that the patient remains stationary with respect to the coordinate system of its built-in infrared camera. Our method thus is a special case of static occlusion handling as described in [2].

The basic idea of static occluding handling is to define a polygonal representation of physical objects. These so-called phantom models are then utilized in the AR rendering pipeline to achieve the effect of virtual objects appearing occluded. Unlike the simple polygonal descriptions used in most applications, models derived from medical volumetric datasets can consist of several million triangles. This problem is aggravated by the steadily increasing resolution provided by modern medical scanners. In order to prevent occlusion handling from having an exceedingly negative impact on the overall frame rate, we have thus devised a method for reducing the number of triangles.

2.1 Volume Preprocessing

We describe a volume preprocessing pipeline for reducing the polygonal complexity of models generated from medical datasets. Unlike general simplification schemes, our method tries to preserve a highly detailed model of the anatomy’s outer surface, while completely eliminating inner structures invisible from the outside. This is achieved by extracting the visual hull volume, a binary volume in which all parts of the volume dataset than can not be seen from the outside, including anatomical cavities, have full intensity. In order to compute this visual hull volume, we perform a number of first-hit raycasting tests from six orthogonal directions. This can be considered a simple approximative simulation of possible outside views of the volume dataset. An example of a visual hull volume is shown in Figure 3.

2.1.1 First-Hit Raycasting

The elementary operation of our algorithm is the casting of a single ray parallel to one of the coordinate axes into the volume dataset. The ray is iteratively traversed until a voxel intensity above a user-defined threshold is encountered. Until that threshold condition is met or the ray leaves the volume boundaries, zero values are written into a second volume dataset at the same coordinates. This resulting volume, the visual hull volume, is initialized with full voxel intensities (255 in case of 8-bit volumes) beforehand. This raycasting process is an approximative simulation of the user viewing the outer surface of the volume dataset from one direction.

Parallel rays are generated over the entire area of each of the faces of the volume boundary. The visual hull volume is initialized with full intensities only at the very beginning

of the process. Each of the raycasting iterations then works on the already modified volume. Thus the empty parts surrounding the actual anatomy are ”carved out” of the volume consecutively. This principle is shown in the following piece of pseudocode:

```
originalVolume: VolumeData;
visualHull:      VolumeData;
rayCastingDirs: Vector3d[6] :=
  {{ 0, 0, 1}, { 0, 0,-1}, { 0, 1, 0},
   { 0,-1, 0}, { 1, 0, 0}, {-1, 0, 0}
  };

procedure preprocessVolume(threshold: double)
startPos, pos: Vector3d;
begin
  initializeWithFullIntensity(visualHull);

  for directionLoop := 1 to 6 do
    for startPos ∈ all positions on corresponding
      face of volume boundary do
      pos := startPos;

      while (pos within volume boundaries) do
        if (originalVolume[pos] > threshold)
          break;
        visualHull[pos] := 0;
        pos := pos + rayCastingDirs[directionLoop];
      done

    done
  done
end
```

While the first-hit raycasting method works well for most volume datasets, ray iterations can be terminated erratically in cases with an above-average level of noise. We have thus embedded the raycasting process into a volume processing pipeline comprising additional filtering steps.

2.1.2 Volume Processing Pipeline

Before the actual raycasting operation is applied to the volume dataset, two filtering steps are performed. The first is a standard low-pass Gaussian volume filter, which reduces noise in the volume. Like for most steps in our volume processing pipeline, the user can opt for skipping the Gaussian filter step. If Gaussian smoothing is applied to the dataset, the standard deviation used by the filter can be modified by the user. After this low-pass filtering step, morphological operations can be applied to the volume, which remove isolated islands of low or high intensity. The kernel size used for the morphological opening and closing steps can be adjusted manually.

The computation of the visual hull volume from the preprocessed volume dataset as described in Section 2.1.1 only yields voxels with either full intensity or zero intensity. Since such a quasi-binary volume dataset leads to visible artefacts in the subsequent polygonal iso-surface extraction, another volume filtering step is applied beforehand. In order to remove possible remaining noise and smooth the edges, first a median filter and then another Gaussian smoothing are used. Unlike for previous steps of the pipeline, we have found fixed kernel sizes to work well for the postprocessing of visual hull volumes. The median filter uses a kernel size of 3x3x3 voxels, while the Gaussian postprocessing filter has a standard deviation of 1.5 voxels. The result of postprocessing with a Gaussian filter operation is illustrated in Figure 1.

If areas of full voxel intensity are located at the boundary of the volume dataset, the subsequent iso-surface extraction

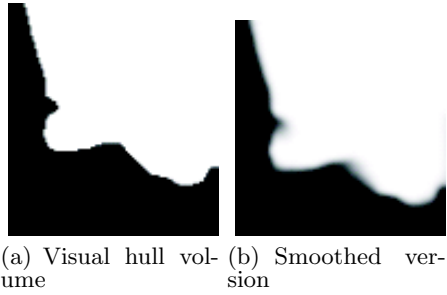


Figure 1: After the raycasting process, the visual hull volume is smoothed using a Gaussian filter. A magnified detail of one slice of a visual hull volume is shown.

is unable to generate a suitable polygonal representation. In such cases, the computed iso-surface will have holes at the respective places of the visual hull volume. The user can thus select to apply a final postprocessing step to the volume, in which it is padded with a surrounding layer of zero intensity voxels. This additional layer causes the iso-surface extraction algorithm to generate a closed outside surface for the whole volume. Figure 2 gives an overview of the entire volume processing pipeline used for computing the visual hull volume.

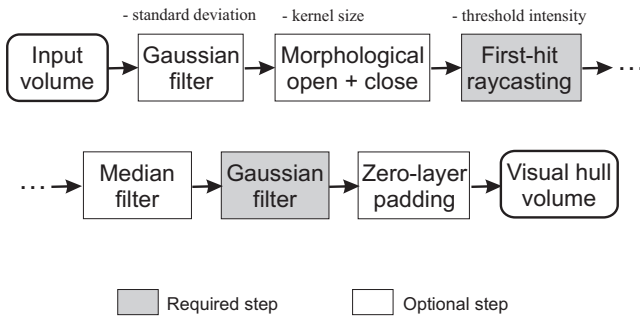


Figure 2: Overview of the volume processing pipeline involved in computing the visual hull volume. The steps in boxes with white backgrounds are optional. The items above some boxes list user-definable parameters.

2.2 Integration into Rendering Process

A polygonal iso-surface is extracted from the visual hull volume using the standard marching cubes algorithm. Since the visual hull volume contains a representation of the outer surface as the interface between full and zero intensity, the half of the full intensity is selected as iso-surface threshold (128 in case of 8-bit volumes).

This visual hull iso-surface is then used for suppressing the display of occluded graphical objects during the augmented reality image composition process. In order to achieve this effect, the iso-surface is rendered only into the Z-buffer (depth buffer) before the actual virtual objects. When the graphical objects are rendered, their color values overwrite the original camera image, but pixels lying behind the (invisi-

Table 1: Triangle counts in iso-surfaces generated from unprocessed original and visual hull volumes

Dataset	Original	Visual hull	Reduction
Plastic skull	2,219K	1,279K	42.4%
Patient A	3,312K	1,627K	50.9%
Patient B	767K	230K	70.0%
CTHead	339K	283K	16.5%
MRBrain	433K	282K	34.9%
Average			42.9%

ble) visual hull are suppressed. Since the graphical model contained in the volumetric dataset corresponds to actual patient anatomy, the impression of virtual objects being occluded by the patient is created.

3. RESULTS

We have tested the visual hull volume extraction with several datasets. Five different medical volume datasets were used for the evaluation. Figure 3 shows one slice of an example patient dataset for the original input (Fig. 3(a)) and computed visual hull volume (Fig. 3(b)). As illustrated in the figure, the visual hull volume generated by our pipeline is an accurate representation of the outer surface of the patient’s anatomy.

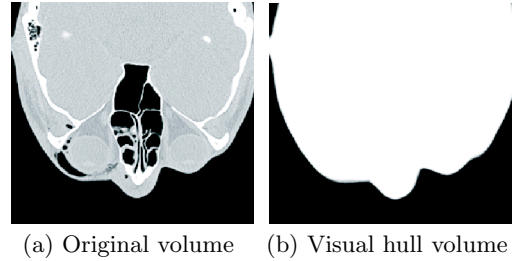
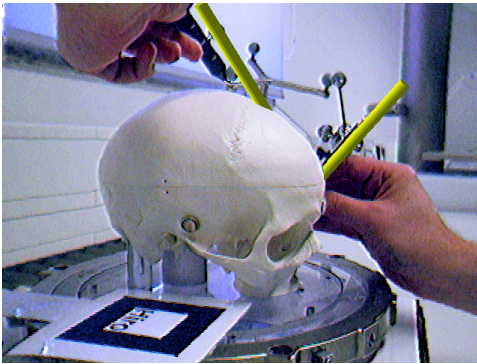


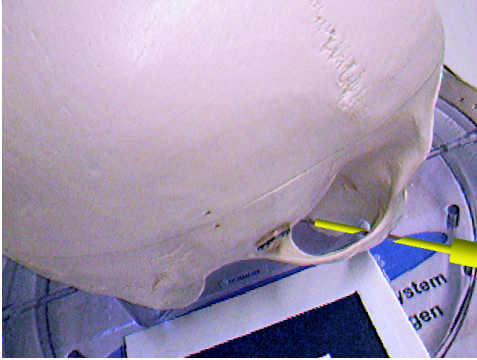
Figure 3: Visual hull volume computed for an example dataset. Both images show the same volume slice.

We have extracted iso-surfaces for both the original input and the visual hull volumes of all five test datasets. A comparison of the triangle counts of the generated surfaces is shown in Table 1. As shown in the table, a triangle count reduction of at least one third is achieved in all but one cases. While reduction rates of close to or over 50% frequently occur, only dataset “CTHead” is not significantly simplified by our volume processing pipeline, due to the already low complexity of the input volume.

The effect of our occlusion handling method on augmented images is shown in Figure 4. The long cylindrical shapes are virtual graphical objects representing surgical tools. These graphical models are positioned corresponding to actual surgical tool mock-ups held by the user. Figure 4(a) shows that using our method, the surgical tool models are correctly occluded by the skull. Much of the geometry cannot be seen because the rendering of the respective pixels is suppressed by the depth values of the visual hull surface of the skull in the depth buffer. The fact that small details in the medical dataset are preserved by our volume preprocessing pipeline is illustrated in Figure 4(b), in which the surgical tool model is partly occluded by the cheek-bone.



(a) Occlusion handling for multiple tools



(b) Cheek-bone detail

Figure 4: *The effect of our occlusion handling method in the augmented reality setup. Graphical representations of surgical tools are used as virtual objects.*

In order to examine the benefits of our visual hull volume extraction algorithm, we compare the times required for rendering iso-surfaces generated from unprocessed and processed volumes. Table 2 lists average rendering times measured during test sessions. The tests were performed on a high end computer system with an Intel Pentium 4 Xeon processor and an NVidia GeForce FX 5900 graphics chipset. While some of the absolute speedups listed in Table 2 may not seem very large, it has to be noted that rendering iso-surfaces comprising several 100K triangles takes considerably longer on most slower computer systems. Still, even in our tests the absolute reductions of ca. 20 milliseconds achieved for the datasets “Plastic skull” and “Patient A” are a significant improvement. This is confirmed by a perceivable increase of the overall frame rate when using visual hull volume iso-surfaces instead of the original volume for a larger dataset. As shown in Table 2, in terms of relative speedup we have generally measured considerable performance improvements through our algorithm.

4. CONCLUSION

We have presented a new method for handling occlusion in a medical augmented reality system. While the volume processing pipeline was designed for volumetric datasets containing patient anatomy, we believe that our approach can easily be employed for other applications in which volume datasets are available. These could include fields like engineering or production planning. Our tests have shown that

Table 2: Time required for iso-surface rendering

Dataset	Original	Visual hull	Speedup
Plastic skull	35.7 ms	17.4 ms	105.2%
Patient A	51.7 ms	23.3 ms	121.9%
Patient B	14.4 ms	6.4 ms	125.0%
CTHead	4.6 ms	3.7 ms	24.3%
MRBrain	5.9 ms	4.9 ms	20.4%
Average			79.4%

our method is capable of generating easily understandable augmented images, in which virtual objects appear correctly occluded by the patient anatomy.

The visual hull volume extraction process aims at ensuring that the occlusion handling does not have a negative impact on the frame rate. To achieve this, an iso-surface is generated, in which all inner structures are removed since they do not play a role for the occlusion handling algorithm. It is conceivable that our first-hit raycasting strategy with rays parallel to coordinate axes might not work optimally for certain volumes, but we have found it to generate very good results for all our test datasets.

The occlusion handling method described in this paper has been integrated into our medical augmented reality system *ARGUS*. *ARGUS* is designed to depend only on existing and commercially available medical equipment. The improved visual quality achieved through our occlusion handling method could further promote the acceptance of augmented reality in medicine.

This work has been supported by project VIRTUE in the focus program on “Medical Robotics and Navigation” of the German Research Foundation (DFG).

5. REFERENCES

- [1] R. Azuma. A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, 1997.
- [2] D. E. Breen, R. T. Whitaker, E. Rose, and M. Tuceryan. Interactive Occlusion and Automatic Object Placement for Augmented Reality. *Computer Graphics Forum*, 15(3):11–22, 1996.
- [3] J. Fischer, H. Regenbrecht, and G. Barattoff. Detecting Dynamic Occlusion in front of Static Backgrounds for AR Scenes. In *Eurographics Workshop on Virtual Environments (EGVE)*, pages 153–161, May 2003.
- [4] G. Gordon, M. Billingham, M. Bell, J. Woodfill, B. Kowalik, A. Erendi, and J. Tilander. The Use of Dense Stereo Range Data in Augmented Reality. In *Proceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, September 2002.
- [5] V. Lepetit and M.-O. Berger. A Semi-Automatic Method for Resolving Occlusion in Augmented Reality. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2000.
- [6] S. Malik, C. McDonald, and G. Roth. Hand Tracking for Interactive Pattern-based Augmented Reality. In *Proceedings of IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, September 2002.
- [7] M. Wloka and B. Anderson. Resolving Occlusion in Augmented Reality. In *Symposium on Interactive 3D Graphics*, pages 5–12, 1995.