Andreas Wachter*, Jan Kost, and Werner Nahm

MATLAB Simulation Environment for Estimating the Minimal Number and Positions of Cameras for 3D Surface Reconstruction in a Fully-Digital Surgical Microscope

https://doi.org/10.1515/cdbme-2018-0124

Abstract: Contemporary surgical microscope systems have excellent optical properties but some desirable features remain unavailable. The number of co-observers is currently restricted, by spatial and optical limitations, to only two. Moreover, ergonomics poses a problem: Current microscope systems impede free movement and sometimes demand that surgeons take uncomfortable postures over long periods of time. To rectify these issues, some companies developed surgical microscope systems based on a streaming approach. These systems remove some of the limitations. Multi-observer positions, for example, are not independent from each other, for example. In order to overcome the aforementioned limitations, we are currently developing an approach for the next generation of surgical microscope: Namely the fully digital surgical microscope, where the current observation system is replaced with a camera array, allowing real-time 3D reconstruction of surgical scenes and, consequently, the rendering of almost unlimited views for multiple observers. These digital microscopes could make the perspective through the microscope unnecessary allowing the surgeon to move freely and work in more comfortable postures. The requirements on the camera array in such a system have to be determined. For this purpose, we propose estimation of the minimal number of cameras and their positions needed for the 3D reconstruction of microsurgical scenes. The method of estimation is based on the requirements for the 3D reconstruction. Within the MATLAB simulation environment, we have developed a 3D model of a microsurgical scene, used for the determination of the number of required cameras. In a next step a small, compact and cost-efficient system with few opto-mechanical components could be manufactured.

Keywords: surgical microscopy, 3D reconstruction, camera array, minimal number, simulation, MATLAB, fully digital, framework

1 Motivation

Carl-Olof Nylen’s first use of a simple monocular microscope for surgery on a patient in 1921 and Holmgren’s first surgical application of a binocular microscope in 1923 have revolutionized surgery and led to the rapid spread and development of microsurgical procedures throughout a variety of surgical disciplines [1, 2, 3]. Today surgical microscopes are the gold-standard in microsurgery applications. Recent developments improve the usability and quality of surgical microscopes and also include the combination of different visualization techniques within the surgical microscope system [4], which is in line with the general trend towards augmented reality in surgical applications. Despite all the benefits they offer, contemporary surgical microscopes still suffer from a few fundamental issues. Firstly, the microscope itself has to be positioned between the surgeon and the patient, thus impeding his mobility. Surgeons also complain about uncomfortable postures that have to be taken over long periods of time in certain applications. Secondly, preoperative data from magnetic resonance imaging or computer tomography which are 3D data sets must be converted into an 2D data sets that they can overlaid over the real scene in a surgical microscope. The conversion of the data are computationally relatively expensive and they aren’t still have a good accuracy. These limitations led to the concept of a fully digital microscope, where the optical observation systems are replaced by an array of cameras. From the images of the cameras, a 3D reconstruction of the scene could be generated in real time. Rendered images of this reconstruction could then be supplied to users on different image display devices. Such a system provides an independent visualization device from the optical imaging device this allowed the surgeon an new dimension of freedom during an operation. The system also provides a real 3D data set of the surgical scene in which preoperative data can be merge be a data fusion. First, to develop such a system a core issue must be answered: "How many camera are need to reconstruct an complete surgical scene in 3D"? To answer that question we present a MATLAB simulation framework in this paper with that the question could be answered to answer that question.
2 Methods

The simulation setup based on the principal from a fully digital surgical microscopes which was described from Marzi et al. [6]. For the simulation framework was made some simplification. First, the camera are implemented as pinhole cameras. The angle of aperture aren’t limited also a deep of field wouldn’t be implemented. Second, the common many object (CMO) would be modulate be a simplification. The confocal characteristics of the CMO would be implemented by arrange the camera on a hemisphere with a tilt that all cameras would focus in one reference point. The radius of the hemisphere correspond to the working distance of the surgical microscope as shown in figure 1.

Fig. 1: 3D plot of a camera setup consisting of six cameras. It shows the camera aperture positions as well as the individual coordinate system axes. All six cameras look at the center of the sphere, which corresponds to the reference point.

2.1 Input Parameters

Beside the working distance and the angle of the tilt from the camera the baseline of the cameras are the main input parameters for the camera setup. The relationship between the parameters are illustrated in figure 2

The other input data is a non-manifold triangulated surface mesh from a surgical sites. For testing our simulation framework we chose to approximate structures commonly targeted or encountered in microsurgery, especially in approaches to the brain. These structures are: Blood vessels, nerves – most prominently the cranial nerves that sometimes span across the entire incision channel – tumors and aneurysms. A constellation of vessels and nerves of different diameters and a representation each, for both aneurysms and tumors. We placed this simple anatomical model in a deep operating channel. The complete model is shown in figure 3.

2.2 Vertex Visibility

The first step after import the mesh is the decide for every if a vertex are visible. For this purpose the program iterate through the k cameras and find out how many vertices of each face in the mesh every individual camera could see. At the beginning of each iteration a backface culling to thin out the set of m potentially visible faces as much as possible.

Backface Culling

A solid object has a gapless surface and a defined outside direction. If the object is modeled using a mesh, those traits are tantamount to the mesh being both manifold and consistently oriented with face normals pointing outwards. In addition, the back of an opaque, solid object is always hidden from direct sight of an observer, as the object itself occludes the corresponding view rays. These two considerations lead to the concept of backface culling: When rendering an image of a mesh representative of such an opaque, solid object, it is valid to discard (cull) all faces that are oriented away from the observer (backfaces), as they are definitely invisible. The view ray from the observer to any point on a backface will always be blocked by a face oriented toward the observer (frontface). Faces on the contour are also not visible as they are exactly
copolitan with the observer’s view direction. The set of faces visible from any angle will hence always consist exclusively of frontfaces. These findings can be collected into the following statement: One can safely cull backfaces and any face on the contour from the set of potentially visible faces for rendering, provided that the mesh is manifold and consistently oriented. Before culling, the faces must first be sorted into the three classes according to their orientation: Frontfaces, backfaces and faces on the contour. In order to find the orientation, one must calculate the scalar product \( \overrightarrow{CV} \cdot \overrightarrow{n} \) of the vector describing the view ray \( \overrightarrow{CV} \) with the face’s normal vector \( \overrightarrow{n} \). The view ray is the vector \( \overrightarrow{V} - \overrightarrow{C} \) from the observer \( \overrightarrow{C} \) to any one of the face’s vertices \( \overrightarrow{V} \). Determining the orientation of a face relative to the observer then becomes a matter of examining the value of the scalar product. The three possible cases are listed in Table 1.

<table>
<thead>
<tr>
<th>Scalar Product</th>
<th>Face Orientation</th>
<th>Visibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \overrightarrow{CV} \cdot \overrightarrow{n} &gt; 0 )</td>
<td>frontface</td>
<td>visible</td>
</tr>
<tr>
<td>( \overrightarrow{CV} \cdot \overrightarrow{n} &lt; 0 )</td>
<td>backface</td>
<td>invisible</td>
</tr>
<tr>
<td>( \overrightarrow{CV} \cdot \overrightarrow{n} = 0 )</td>
<td>neither ( \Rightarrow ) face is on contour</td>
<td>invisible</td>
</tr>
</tbody>
</table>

**Tab. 1: Face orientation derived from the value of the scalar product of the viewing direction \( \overrightarrow{CV} \) and the face’s normal vector \( \overrightarrow{n} \).**

**Plücker’s Coordinates**

For each of the remaining non-culled faces (frontfaces), the function then found out if any of the view rays from the current camera aperture to the face’s vertices were occluded. This occlusion check was done using the ray-triangle intersection test in Plücker coordinates [6, 7]. In 3-dimensional euclidean space, a line directed from a point \( P \) through a point \( Q \) can be described as the set of points \( L = \{ P + t(Q - P) \mid t \in \mathbb{R} \} \). Using this parametrization is not advantageous for line-line intersection testing, however, as the test then involves solving a system of linear equations and a case differentiation. Plücker’s coordinates provide an alternative description for directed lines that allows us to circumvent handling equation systems and do the incidence test by computing only one inner product. We even gain information about the spatial relationship between the lines, i.e. on which side of each other they lie. This synergistic side-effect makes it possible to expand the line-line intersection test into a line-polygon intersection test, using the same inner products. If the view ray from the current aperture to the vertex in question intersected any face between the aperture and the vertex, the vertex was certainly occluded. The program only checked for occlusion by other frontfaces, ignoring faces on the contour and backfaces.

### 2.3 Determination of the Reconstructable Area

The reconstructability simulation is the core element of the methods used in this paper. Taking a triangular mesh model of a surface as a basis, the fundamental goal of the simulation is to determine which part of the surface of a given reference model could be reconstructed from the combined image set of all the cameras in an given camera array. The area of the resulting partial surface is then to be calculated. Achieving this goal involved making visibility and reconstructability decisions for each face. Our simulation make each of the decisions relying on a respective assumption. The reasoning and considerations behind the two assumptions:

1. A face is reconstructable from only two camera views. In computer graphics, points on the surface of an object can be reconstructed in 3D by triangulating their position from a set of stereo images of the object. This is done by finding structures that occur in each of the images and then comparing the color values of the pixels in those regions to each other. If the values of a pair of pixels are similar enough, the corresponding point can be reconstructed. The positional uncertainty of the reconstruction through triangulation decreases with an increasing amount of images from mutually different angles. Consequently, the number of cameras to which a face in question is visible allows conclusions about the accuracy with which the position of surface points on that face could be reconstructed, from the images of the camera array. It is hence valid to chose a threshold number of cameras a face has to be visible in, for it to be considered reconstructable with satisfactory accuracy. The minimum possible threshold is two, as the position of a point in 3D cannot be reconstructed from just one 2D image.

2. A face is visible if at least \( v_{\text{min}} \) of its vertices are visible. The simulation inferred a face’s visibility to a given camera from the number \( v \) of it’s vertices the camera could see. The method of inference was to compare \( v \) to a threshold, which we hence refer to as the visibility or vertex threshold \( v_{\text{min}} \). A face was treated as visible if \( v_{\text{min}} \) or more of it’s vertices were visible to the camera. There are four possible values for \( v_{\text{min}} : 0, 1, 2, 3 \). In our simulation program, a face was treated as definitely invisible if none of its vertices were visible.

### 3 Results

The MATLAB simulation framework provides with a given input mesh, a camera setup and a vertex threshold two different outputs. First it provides a 3D model. This is shown in figure
4, where 6 single output models would be combined together in one. The numbers of the cameras are coded in different colors. Second, the size of the reconstructable area in \( \text{mm}^2 \). The reconstructable area for different vertex thresholds and cameras are plotted in figure 5.

### Fig. 4
This figure shows which additional areas of the model become reconstructable when the number of cameras is increased. 2 cameras: grey and transparent parts; 3 cameras purple; 4 cameras: orange; 8 cameras: blue; 16 cameras: yellow; ground truth: green patches.

### Fig. 5
This figure contains a scatter plot of the data points for 2, 3, 4, 8, 16 and ground truth, colored according to the underlying vertex threshold. The curves are function graphs for the generalized hyperbola fit functions.

## Discussion and Conclusion

The simulation framework which we presented in this paper is first approach to answer the initial question: "How many cameras are needed to reconstruct a complete surgical scene in 3D?" The input parameter for framework are only the camera setup and a triangle mesh. The Framework don’t need color information like other rendering software. It only uses structural information with backface culling and Plücker coordinates to make the reconstructability decision. The reason why we don’t use commercial rendering software is that we won’t have to deal with lighting and color information. We are only interested in the first in the structural surface information of the model. Now, we have a pure knowledge to answer the initial question. This answer couldn’t yet be final given, because there are still some limitations and simplifications in the simulations framework. But the tool allows to approximate the minimal number of cameras by now. A second open issue is the ground truth data. Actual the ground truth data are calculated for a stereo pair of camera which is rotated in 1 degree around the model with the same baseline as a commercial microscope has.

## Outlook

The next development steps to tackle the open issues and limitations are the implementation of a Field of View for the cameras. This step as an direct influence reconstructable area. To increase the accuracy partially hidden faces must be remeshed, so that no part of a visible face would be occluded. We hope with that improvements we can find answer the initial question.

### Author Statement

Research funding: Mr. Andreas Wachter is funding by the state of Baden-Württemberg. Conflict of interest: Authors state no conflict of interest. Informed consent: Informed consent is not applicable. Ethical approval: The conducted research is not related to either human or animals use.

### References