Incremental Visual Hull Reconstruction

Ali Bigdelou
http://campar.in.tum.de/Main/AliBigdelou
Alexander Ladikos
http://campar.in.tum.de/Main/AlexanderLadikos
Nassir Navab
http://campar.in.tum.de/Main/NassirNavab

Chair for Computer Aided Medical Procedures
Technische Universität München
Boltzmannstr. 3, 85748 Garching, Germany

Visual hull reconstruction is the process of generating three dimensional scene geometry from silhouette images captured from different views of a scene. The visual hull is the shape maximally consistent with its silhouette projection [4]. It has been popular due to its good approximating qualities for many objects and its ease and speed of implementation. It is commonly used as a starting point for more elaborate 3D reconstruction methods [5] and finds application in many real-time 3D reconstruction systems due to its good performance [3, 6].

However, current reconstruction systems do not take temporal consistency in the scene into account. In contrast to these approaches, we propose an incremental voxel based visual hull reconstruction method for dynamic scenes using ray casting. By exploiting the fact that there usually occur only small changes between consecutive frames, we can reduce the runtime and calculation complexity in these environments. This allows us to perform real-time reconstruction on a standard processor without having to parallelize the computations.

In dynamic scenes such as those captured by multi-camera systems the changes between frames are limited by the speed at which the objects in the scene move or deform. Hence, the number of voxels that change in the reconstruction does not alter dramatically. Therefore it is inefficient to reconstruct each frame independently. Theoretically the only voxels that have to be updated to transform the previous reconstruction into the current one, are the ones with changed occupancy.

Here, we explain our algorithm for incremental visual hull reconstruction in dynamic scenes, which uses ray casting to update the visual hull. The first step for updating the reconstruction is to compute the difference images from silhouettes. This allows us to determine which pixels were added or removed from the foreground. Then for each changed pixel in the difference images, we create a ray from the optical center of the camera passing through the pixel and traverse all voxels, which the ray passes, using ray casting [1].

Algorithm 1 Pseudo code for the deletion phase in the ray casting method

```
1: for each image i in the sub-sampled difference images list do
2:     for each removed pixel p in image i do
3:         Create ray r from optical center passing through p
4:         Set r to the first voxel in the reconstruction volume
5:         repeat
6:             Set occupancy state of the voxel at r to empty
7:         until step r to the next occupied voxel is unsuccessful
8:     end for
9: end for
```

In the deletion phase, the occupancy of all voxels lying on rays passing through removed pixels is set to empty. This is because these areas are the projection of removed voxels since the previous time step. This is illustrated in figure 1. Algorithm 1 shows the pseudo code for the deletion process. To complete the updating process, after removing deleted voxels, newly occupied voxels are added in the addition phase. Figure 1 shows the process of creating newly occupied voxels in the reconstruction results of the previous phase. The addition phase of this method is similar to the deletion phase. However, before creating new voxels, their occupancy has to be checked.

Although the ray casting approach described in here is already very efficient, we can further improve the performance by storing more data from the previous time step. To this end we store per pixel ray information as described in [2] for each time step. We save the basic ray information such as starting point, direction and current voxel for later use in successive time steps. We call this stored information ray buffers. This information helps us to define an even smaller search space.

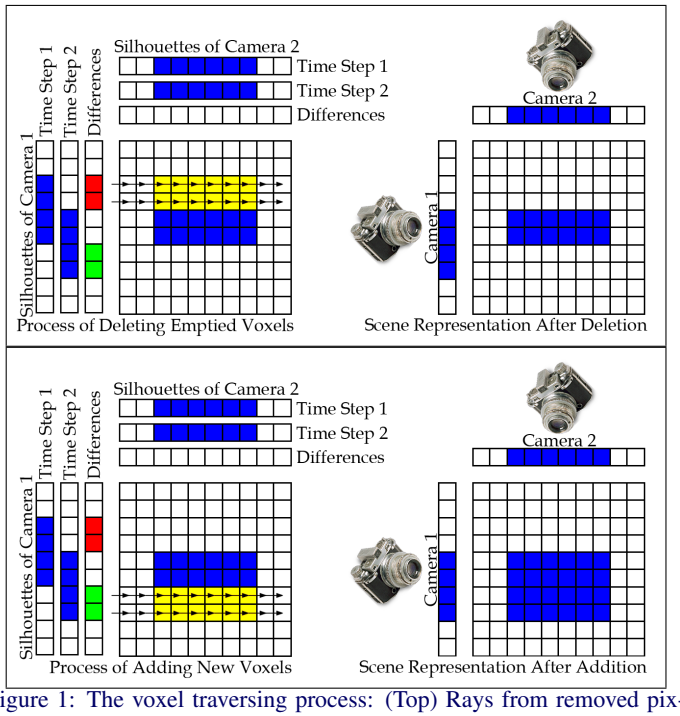

Figure 1: The voxel traversing process: (Top) Rays from removed pixels. Yellow voxels are removed; (Bottom) Rays from new pixels. Yellow voxels are added.

Our proposed approaches update the scene based on the changed areas in the images, so each change in the visual hull of a model should be reflected in at least one of the silhouette images used for reconstruction. We have shown the correctness of this assumption through mathematical analysis. In other words, we have shown that for any newly removed (added) voxel from the scene at time $t+1$, which alters the visual hull, there exists at least one difference image, that reflects this change.

We have successfully tested our proposed incremental reconstruction approaches as well as previous methods on both synthetic and publicly available real data sets, comparing them on different aspects such as run time and the number of occupancy checks with various voxel and image resolutions. These results show the significantly improved performance of our incremental approaches. We achieve a speed-up of up to 10 times over the other reconstruction methods.

[1] J. Amanatides and A. Woo. A fast voxel traversal algorithm for ray tracing. In *Eurographics '87, Amsterdam, North-Holland*, 1987.

[2] O. Batchelor, R. Mukundan, and R. Green. Ray casting for incremental voxel colouring. In *Image and Vision Computing Conference - IVCNZ05 NewZealand*, 2005.

[3] A. Ladikos, S. Benhimane, and N. Navab. Efficient visual hull computation for real-time 3d reconstruction using cuda. In *Proceedings of the 2008 Conference on Computer Vision and Pattern Recognition Workshops*, 2008.

[4] Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE PAMI*, 16(2):150–162, 1994.

[5] S. Seitz, B. Curles, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE CVPR*, 2006.

[6] L. Soares, C. Ménier, R. Raffin, and J. Roch. Parallel adaptive octree carving for real-time 3d modeling. In *IEEE Virtual Reality*, 2007.