Efficient Voxel Marking for Hierarchical Volumetric Fusion

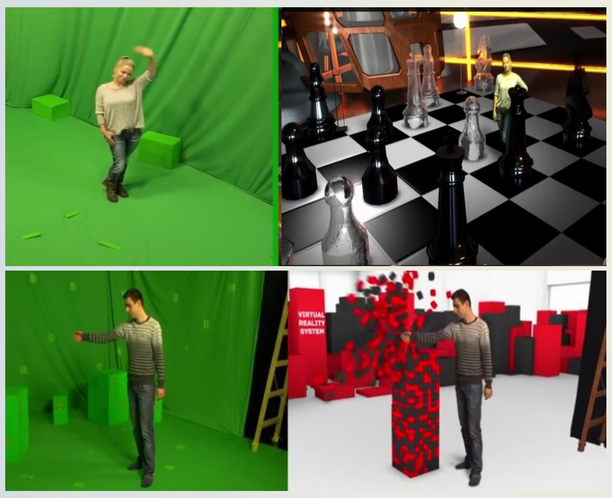
László Szirmay-Kalos, Balázs Tóth, and Tamás Umenhoffer

Dept. of Control Engineering and Information Technology, Budapest University of Technology and Economics, Hungary

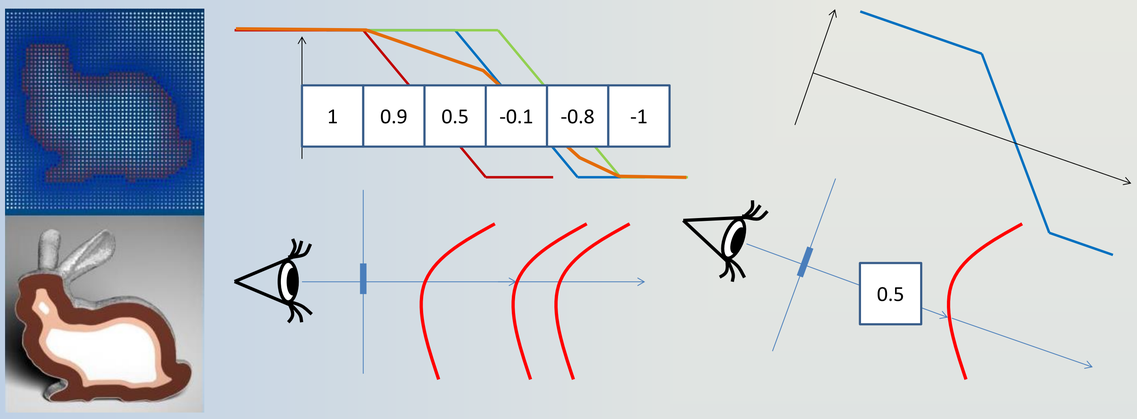
Abstract: When fusing depth images into a 3D volumetric model, a crucial task is to mark macro-cells as empty or as intersected by the noisy surface represented by the depth image. This paper proposes a simple marking algorithm for the GPU implementation of hierarchical volumetric fusion. The method is based on multi-level DDA ray-casting. The GPU implementation is of scattering type, but we also show a solution to avoid atomic writes, which improves performance.

Motivation: Real-time volumetric fusion of 3D depth images

Zinemath RGBZ camera system



Volmetric fusion of 3D depth images

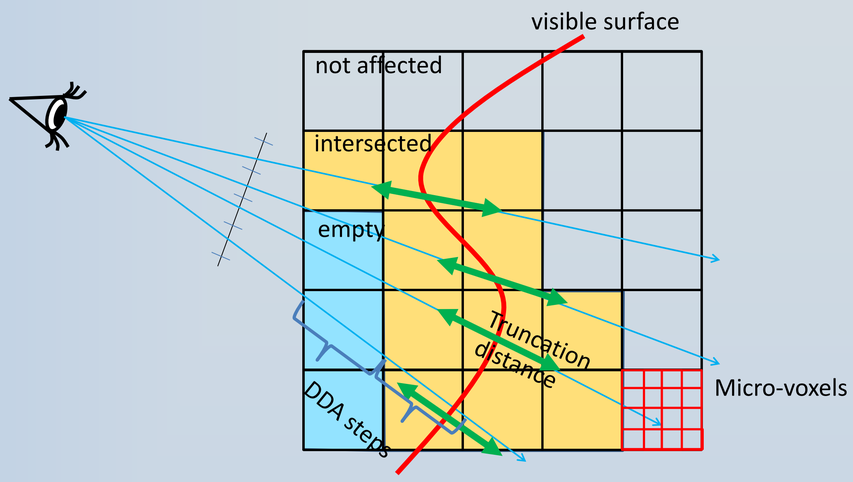


3D object is reconstructed from depth images as a 3D Truncated Signed Distance Field (TSDF).

Volumetric fusion repeats the following steps:

1. **Fusion**: Find cells that are empty or affected by the current depth image and fuse, i.e. average their stored TSDF value with the TSDF obtained from the current depth image.
2. **Rendering**: Execute ray casting to render the surface from the current camera also computing the surface normals.
3. **Get depth image**: Read the new depth image from a possibly moving camera and compute the normals of the back projected depth image.
4. **Camera tracking**: Based on the rendered surface and normal vectors and the measured distance values, compute the new camera position and orientation with the Iterated Closest Point (ICP) algorithm

Hierarchical fusion to attack Step 1



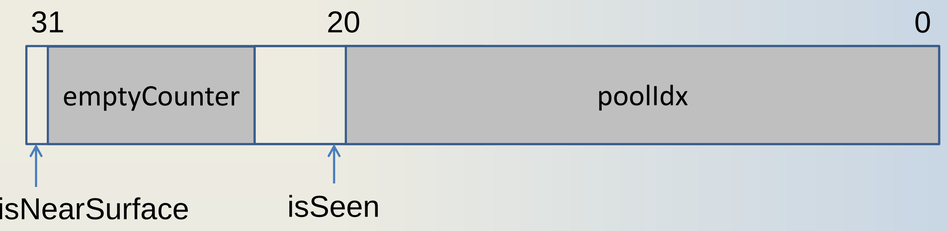
Classical solution: The center of the voxel is projected on the window plane of the depth image to locate the pixel where it is visible from the depth camera. If the difference of the depth value and the distance of the voxel center from the camera is less than the truncation distance, the voxel is affected, otherwise it is assumed not to be intersected by the currently visible surface. This method works only if the voxels are small and they are projected to a single pixel, otherwise sampling artifacts may show up.

Scalable Kinect Fusion solution: Hexagons of each voxel are projected onto the window plane and conservatively rasterizing the projected polygon to identify the pixels where depth comparison is needed. If rasterization is done in parallel, then an additional reduction is needed to make the final conclusion for the hexagons.

Our solution: The marking process assigns a GPU thread to every pixel of the depth image. The thread takes the depth value of the camera in this pixel and forms an interval, where the minimum is the depth value minus the truncation distance, and the maximum is the depth value plus the truncation distance. The thread executes a DDA based voxel traversal method to identify those higher level cells that are intersected by this ray. Until the ray parameter at the exit point of the cell is lower than the minimum value, the visited cells are marked.

Avoiding atomic writes in an input driven, i.e. scattering type GPU algorithm

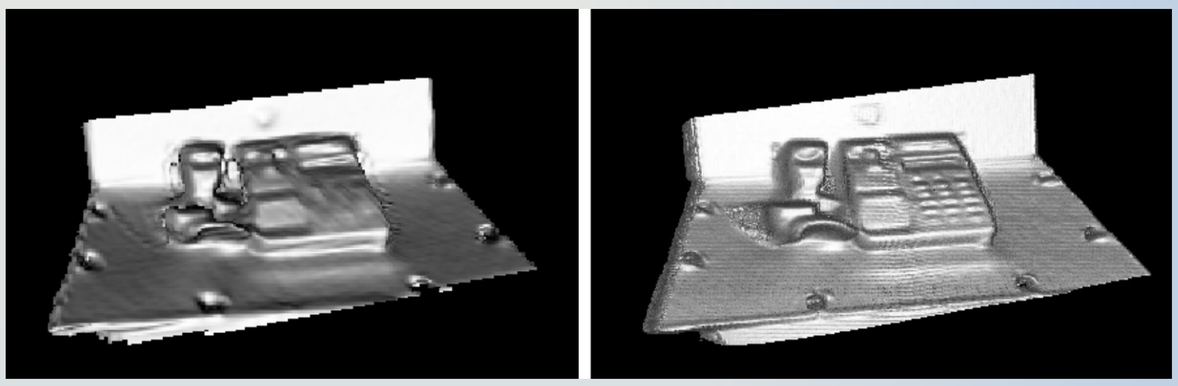
As different rays may intersect the same macro-cell, different threads may update the flags of the same cell, causing write collisions and usually necessitating slower atomic writes. However, in this special case, atomic operations can be saved. The two flags are put into two bytes of the descriptor word, thus each of them can be accessed without modifying the other flag. If needed, a thread sets a flag independently of its previous value, and all other bits of the byte are constant during the execution of this thread. So the result is independent of the order how threads access these bytes.



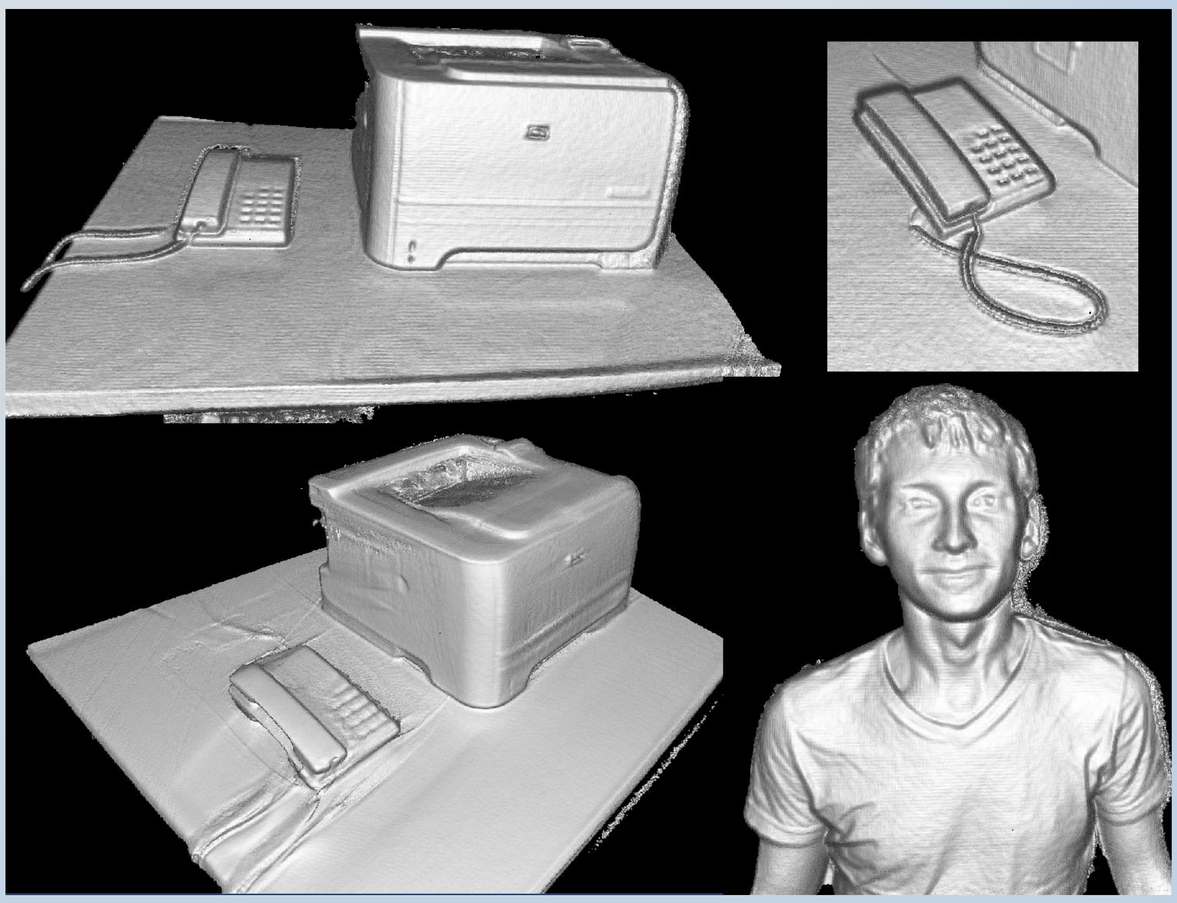
One frame late option

In Step 2 called Rendering, a ray-casting needs to be executed anyway to track the camera by ICP, thus the identification of affected cells can be merged with this step reducing the additional cost of marking to almost zero. However, rendering happens with the old camera position and orientation while the fusion should use the updated camera parameters. As our algorithm marks only macro-cells, the one frame delay does not result in inaccuracies, but it can happen that voxels of a macro-cell are not updated in a frame or are tried to be updated when it is not necessary. Note that ICP requires small camera movements.

Results



Comparison of the commercial version of KinectFusion (left) and the proposed algorithm (right) when the two methods allocate the same amount of GPU memory. Voxel edge length could be reduced from 8 mm to 1 mm.



References

[1] CHEN J., BAUTEMBACH D., IZADI S.: Scalable real-time volumetric surface reconstruction. ACM Transactions on Graphics (TOG) 32, 4 (2013), 113. 1

[2] CURLESS B., LEVOY M.: A volumetric method for building complex models from range images. SIGGRAPH ’96, pp. 303–312.

[3] I ZADI S. ET AL.: 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, UIST ’11, pp. 559–568