1. Abstract

Modern emergence of automation in the industry and everyday life is leveraged by extensive research in mobile robotics. Novel 3D sensors such as laser scanners or cameras enable cars to drive autonomously. UAVs to observe critical environments, or an underwater robot to construct pipelines. However, 3D sensor samples do not provide the intrinsic information a robot needs to operate on. Voxel based shape modelling has been identified as a fruitful solution. However, its application is limited to small areas since processing and visualization of large environments is very challenging. Dense voxel grids allow fast data access but suffer from a large memory overhead. Modelling an area of 100x100x100m with a resolution of 1cm would result in a 3.7TB memory requirement. Motivated by this, sparse voxel octrees (SVO) [4] have been proposed. These however, increase the data access complexity from O(1) to O(d) with d being the depth of the octree. This means, that the larger a scene, the slower the data access. This works proposes a constant data access scheme for huge 3D environments combining hash tables with SVOs.

2. Proposition

We propose to combine octrees with hash tables (Figure 1) leading to sparse voxel representation well suited for efficient storage and fast data access. The hash table is used to access SVO root nodes, which further contain an octree in itself. Since the internal octrees are constructed of small depth (d = 1), this dramatically decreases the access time complexity to O(d). For a standard octree of depth d = 16, this is a speedup of factor 16. The access time comparison is shown in Table 1.

The novel efficient data structure is further applied for incremental 3D modelling from camera based 3D sensor illustrated in Figure 2.

3. Hash-Tables

To access the voxel at index coordinates (x,y,z), we begin by computing the following signed rootKey

\[ \text{int rootKey}(x) = (1 < \log(2x) - 1), \]
\[ \text{int rootKey}(y) = (1 < \log(2y) - 1), \]
\[ \text{int rootKey}(z) = (1 < \log(2z) - 1); \]

where & and * denote, respectively, bit-wise AND and NOT operations. At compile time, this reduces to just three hardware AND instructions. Similar to [1], the rootKey is further processed to a hash.

\[ \text{unsigned int rootHash} = (1 < \log(2N - 1)), \]
\[ \text{rootKey}(0) = 13856093, \]
\[ \text{rootKey}(1) = 19349663, \]
\[ \text{rootKey}(2) = 83492791; \]

Here, N is the constant length of the hash being generated, the three constants are large prime numbers, ^ is the binary XOR operator, & is the bit-wise AND operator and <= is the bit-wise left shift operator. Finally, google::dense_hash_map from [2] is applied to store and search the octree root nodes.

4. Conclusion and Outlook

Voxel based 3D models enable an environment to be approximated from data streams of multiple 3D sensors. Applying the proposed approach, the tremendous amount of data can be fused to a consistent and large 3D model of a city or large countryside. We would like to evaluate the implication of the approach for the security relevant modelling of dynamic environments, where automated monitoring of the condition (e.g. road roughness) and geometrical properties (e.g. obstacles) is of interest.

References


Acknowledgements

This work was supported by the HRSST.

Figure 1: The basic structure of the proposed method combining a hash map for search of the top level octree nodes.

<table>
<thead>
<tr>
<th>Octree</th>
<th>Access Time</th>
<th>Max Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>d=16</td>
<td>6.43μs</td>
<td>32768³ (327m)³ @1cm</td>
</tr>
<tr>
<td>d=16</td>
<td>2.55μs</td>
<td>32768³ (327m)³ @1cm</td>
</tr>
<tr>
<td>Hashed-Octree</td>
<td>0.45μs</td>
<td>∞</td>
</tr>
</tbody>
</table>

Table 1: Access time and resolution performance.

Figure 2: a) Flight over a chapel, b) corresponding voxel based 3D model, c) zoom on the wall pattern of the chapel.