Fast Voxel Maps with Counting Bloom Filters

Julian Ryde and Jason J. Corso

***Abstract*— In order to achieve good and timely volumetric mapping for mobile robots, we improve the speed and accuracy of multi-resolution voxel map building from 3D data. Mobile robot capabilities, such as SLAM and path planning, often involve algorithms that query a map many times and this lookup is often the bottleneck limiting the execution speed. As such, fast spatial proximity queries has been the topic of much active research. Various data structures have been researched including octrees, k-d trees, approximate nearest neighbours and even dense 3D arrays.**

We tackle this problem by extending previous work that stores the map as a hash table containing occupied voxels at multiple resolutions. We apply Bloom filters to the problem of spatial querying and voxel maps for the example application of SLAM. Their efficacy is demonstrated building 3D maps with both simulated and real 3D point cloud data. Looking up whether a voxel is occupied is three times faster than the hash table and within 10% of the speed of querying a dense 3D array, potentially the upper limit to query speed. Map generation was done with scan to map alignment on simulated depth images, for which the true pose is available. The calculated poses exhibited sub-voxel error of 0.02m and 0.3 degrees for a typical indoor scene with a map resolution of 0.04m.

I. Introduction

A good representation of the operating surroundings (map) is vital for many tasks that mobile robots need to accomplish. Examples of such tasks include mapping and localisation (SLAM), path planning, obstacle avoidance, ray tracing and object detection and recognition.

The map building process can be considered to consist of two phases. The localisation of the robot or sensor within the map and the update of the map with this new data. As such the map data structure should support both operations, answering queries and incorporating new data very quickly, all the while maintaining a small memory footprint.

Many representations have been explored including occupancy grids in 2D, point clouds [1], parametric representations such as planes [2] and salient features [3]. Currently the most popular approach for mapping with 3D data and the corresponding 6 degree of freedom poses is point clouds aligned with iterative closest point (ICP, [4]) and a mechanism to accelerate the nearest neighbour lookup on the point clouds.

For fast queries on large point clouds approximate nearest neighbour techniques have recently proven popular, [5]. For even greater query speed researchers have resorted to generating local dense 3D binary arrays in which every voxel is stored. These dense arrays allow map queries at close to the theoretical minimum lookup time. The problem with dense 3D arrays is the prohibitively large memory required for relatively small operating volumes. One gigabyte of memory can only hold a 10m cube at 1cm resolution with one byte per voxel.

Others have implemented tree based volumetric mapping structures such as K-d trees [6] and octrees [7]. Octrees have the advantage of representing free space as well as occupied voxels. For localisation and SLAM the representation of free space is not as important because more information is derived from observing occupied voxels due to their rarity in comparison to free voxels. Distinguishing free space helps with path planning and planning for exploration but can be done at a coarser resolution.

Previous works by the authors describe a memory efficient and highly adaptable map representation framework referred to as the Occupied Voxel List (OVL) and the Multiple-Resolution Occupied List (MROL) [8]. This framework is used for accurate alignment based segmentation akin to 3-dimensional background subtraction in [9]. The individual occupied voxel lists of the MROL are implemented as hash tables with the keys being the integer voxel indices of the nearest cubic lattice point. The hash table based procedure of occupied voxel lists combined with only storing voxels which are occupied reduces memory requirements at the cost of increased query time. The hash tables have constant lookup time with respect to the number of voxels in the map however this can be quite slow mainly due to the handling of hash collisions.

These structures save memory by only encoding occupied voxels in the robot's surroundings and, we will demonstrate, can be made almost fast as look ups in a dense 3D array. Adding a Bloom filter to the lookup procedure makes these accelerations possible.

The previous localisation approach described in [8] is especially effective for global localisation when no initial pose estimate for each scan is available. Similarly, global localisation on a multi-resolution map is tested by [10]. Iterative alignment methods such as ICP cannot work in such situations. However, it is often the case that a pose estimate for each scan is available, particularly relative pose transforms between subsequent scans as acquired by a moving robot with odometry sensors. Therefore it would be beneficial if the same map data structure could also support localisation given an initial guess for each 3D point cloud. Although scan to scan pose estimates can be acquired, it is more accurate to perform scan to map matching. Thus in this work we also implement an iterative gradient descent optimization for scan to map matching given an initial guess.

J. Ryde and J. J. Corso are with Department of Computer Science and Engineering, SUNY at Buffalo, Buffalo, NY, USA {jryde,jcorso}@buffalo.edu

This initial guess can be quite inaccurate, for instance the pose of the previous 3D scan, and still lead to the correct result.

The ramifications of this work are two fold. First the development of an algorithm that allows faster iterative alignment on volumetric maps. This means that the same volumetric map structure can support global as well as local localisation (determining pose when supplied with a good initial guess pose). Secondly, we present for the first time a method involving Bloom filters and packed voxel indices that allows a sparse representation as well as query times approaching the likely minimum possible, that associated with querying a dense 3D array. Applying Bloom filters to accelerate spatial querying could also benefit fields where fast spatial querying is paramount such as collision detection in physics engines, protein folding and computer graphics rendering and ray tracing.

II. Accelerated voxel occupancy lookup

Use of an OVL based map representation is inherently biased towards a high number of lookup operations. These tend to occur during the course of aligning new scan data to the map and for ray tracing operations. The ray tracing is required for calculating free space voxels, needed for segmentation and path planning.

The process steps that dominate the computation time in practical applications are taking a list of points, transforming them (translation and rotation), quantising them to the nearest lattice point and querying the occupancies of the resulting voxels in the map. Whilst well established procedures exist for transformations the lookup operations can take significant time for high numbers of queries, as is common with high density point clouds. The point transformations are easily computed in parallel with consumer graphics cards doing such geometric transforms at very high rates. Executing the map lookups quickly in parallel is much harder because of the widely separated memory access patterns and thus the map queries tend to be the bottleneck in practical implementations. Ideally this lookup time should be fast and independent of map size.

A. Counting Bloom filter for volumetric maps

The Bloom filter, [11], is a memory efficient data structure and associated algorithms that facilitate fast set membership queries in a probabilistic manner. Queries that return absent are guaranteed to be correct but those returning present are correct to a high probability. The probability of false presence depends on the size of the dataset and the Bloom table as given in (1).

The standard Bloom filter consists of a binary array of length m and associated k hash functions. Each hash function converts set elements to a different position in the binary array. When an element is added to the set the bits in the binary array are set corresponding to the results from evaluating the k hash functions. Upon a query the k hash functions are evaluated on the query term and the resulting bit positions checked. If any are not set then the query term is definitely not present in the set. If they are all set then the query term is likely to be present with a small probability of a false positive given by,

$$P_{FP} = \left(1 - \left[1 - \frac{1}{m}\right]^{kn}\right)^k \approx \left(1 - e^{-kn/m}\right)^k. \quad (1)$$

Where n is the number of elements already in the set.

One of the limitations of the original Bloom filter is that it does not allow removal of elements from the set. In real mobile robot applications voxels need to be removed from the map when objects are removed, parts of the environment change or map errors need correcting. As set element removal is required we adopt a counting Bloom filter, [12]. By incrementing at the positions given by the hash functions on insertion into the set, items can subsequently be removed from the set by decrementing at these positions. For guaranteed correct removal the presence of the items should be first confirmed in the hash table with those to be removed that are present passed on for removal from the Bloom filter.

For a volumetric map consisting of occupied voxels the Bloom filter stores the hash bits generated from the set of occupied voxel indices. The actual value (if occupied) can then be looked up in a standard hash table, if required. However for some operations like scan matching this subsequent value retrieval is unnecessary and can be skipped for speed.

B. Circumventing hashing with packed voxel indices

Additionally, the required hashing operation performed on the voxel indices can take some time. To overcome this, the voxel indices are packed into a contiguous region of memory such that the three voxel indices (for 3D space) are packed as four 2 byte integers (with one zero filler) into a single 8 byte representation. The 8 byte integer acts as the voxel key ID, which unlike a normal hash function is guaranteed to be unique. This memory layout avoids hashing the voxel indices to generate an ID for the hash table, accelerating lookup times. Another benefit of this approach over hashing the voxel indices is that such indices can be unambiguously recovered from the voxel ID. The disadvantage is the limit on the voxel index that in this case is restricted to -32768 to 32767 (2 byte integer limit). Practically, this corresponds to a 655m cubed volume at 1cm resolution and this limit can be increased by packing larger integers and accommodating the larger ID. Together these improvements accelerate the lookup process threefold; Fig. 1 contains a more detailed break down of execution time.

III. Scan alignment and map creation

The OVL and MROL frameworks provide a ready means for simultaneous localisation and mapping (SLAM) capabilities in the form of scan to map alignment and incremental map growing. The sampling based approach described in [8] provides a global localisation method for scan to map matching in the MROL framework, whilst iterative methods based on the minimisation of cost functions that characterise the fit of the scan to the existing map (see [9]) provide this functionality where a reasonable pose prior exists for

both the classical OVL representation and those at multiple resolutions. The resolution of the finest occupied voxel list at the zeroth level is ϵ_0 and the resolution of the nth level is $\epsilon_n = \epsilon_0 2^n$.

It has been shown that SLAM conducted in a scan to map and incremental map growing approach leads to slower error accumulation and reduced drift rates compared to standard pair-wise scan matching [13], [14] and the work here uses such an approach. For the small mapping problems considered within this work (data collected over minutes), the place recognition for loop closure constraint and the loop relaxation problems are safely ignored.

Some optimisations to the aforementioned alignment procedures have been developed that not only improve the computational efficiency, but also improve the accuracy of the resultant transforms.

A. Volumetric Sampling

To reduce alignment errors due to range-based point density effects we employ volumetric sampling. The aim is to ensure that the sampled points are both representative of the structure contained within the scan at an appropriate resolution for matching against a voxelised structure and also to reduce the number of points in the scan required in the evaluation of the alignment objective function. An effective means to achieve this is by subsampling the scan, randomly selecting single scan points from each voxel, at a given sample resolution. This volumetric sampling approach differs to the various range-weighting schemes found in the literature [15], [16].

The 3D point cloud arising from a scan in this work is volumetrically sampled to reduce the number of scan points. In our experiments the volumetric sample resolution is adjusted to produce approximately 1000 scans points. Scan to map alignment is performed on these sample points. The good distribution of this sampling mechanism means few sample points are needed for testing candidate poses. Once a good pose alignment is found all the points from the scan are added into the map.

B. Sequential multi-resolution alignment

An improvement to the iterative alignment method described in [9] has been adopted. Rather than calculating the cost function to be optimized as a weighted sum of the hits at the various resolutions, here the iterative alignment is carried out at each resolution from coarse to fine. For the simulated data presented in Section IV-B, the finest resolution ϵ_0 is 0.04m and the multi-resolution map has 4 levels. The number of levels depends on the convergence region required. Larger regions of convergence are required when only poor estimates of initial pose are available. In this work no pose estimate is required as the guess pose is simply the pose of the previous scan.

The selection of voxel size is important for two reasons. As this approach does not perform any geometric characterisation of the environment, the finest voxels have to be large enough to cover gaps in the point cloud due to scan line spacing to prevent alignment to the scan line geometry rather than that of the environment. Secondly, the voxel size is lower bounded by the error on scan points. It is important to appreciate that sub-voxel alignment is possible, for instance with 0.04m resolution pose accuracy of 0.02m and 0.3 degrees was achieved. Such sub-voxel accuracy is possible as the original range data points are transformed first before quantisation for the map intersection calculation. This allows small iterative adjustments to this transformation pose to influence the count of voxelised points against the map (overlap) that defines the cost function.

The sequential alignment procedure is summarised below.

1) Take a volumetric sample of points at the coarsest resolution
2) Iteratively align with a gradient ascent approach to maximise overlap using fixed step sizes related to the resolution of the current level.
3) Use the pose returned at the coarser level as an initial guess for alignment at the next finer resolution.
4) Repeat above until done for the finest level
5) Perform sub-voxel alignment by using all points (rather than a sample) and gradient ascent to maximise overlap using small translations and angles.

This procedure ensures the utmost map quality (minimal blurring/spreading of the map over time) by minimising the number of new voxels that are created to contain the scan points.

The map is stored as a multi-resolution occupied voxel list along with the trajectory consisting of the poses from which the observations were made when generating the map. The advantage of this representation is that it is very memory efficient, because it is only storing occupied voxels, the required memory does not increase significantly with multiple observations. A map encoding the free space can be easily generated dynamically via a sensor model at the recorded observation poses.

IV. Experiments and Results

We have conducted a suite of experiments to evaluate our primary claim: that the use of Bloom filters for volumetric SLAM is faster than currently used methods, such as hash tables, and yet remains accurate and is memory efficient. All experiments with timing information were executed with a single core of an Intel(R) Core(TM) I5 750 running at 2.67GHz with 4GB of RAM. In our first experiment, we evaluate the execution times along the critical path of execution — the process of looking up a point in the map — for implementations with and without Bloom filters.

In our second experiment, we use simulated data generated from a 3D graphics program and assess the overall accuracy of our system with and without the Bloom filters; the advantage of simulated data is that access to the true values of variables such as pose is possible allowing quantitative and systematic evaluation. In our final experiment, to allow for comparative evaluation and reproducible results, we select a benchmark dataset (*thermolab* [17]) that consists of real-world data from an imaging depth sensor. With this real

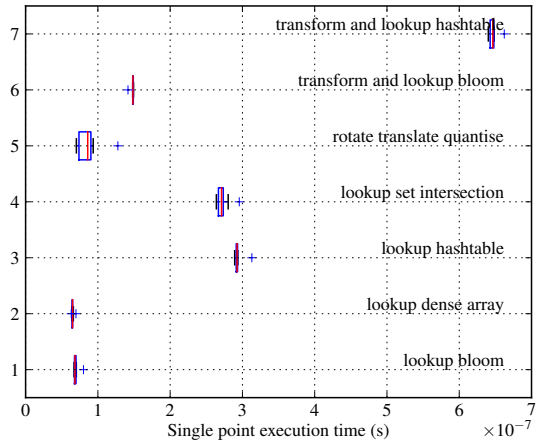

Fig. 1. Box and whisker plot (10 runs) of single core execution times for stages of determining whether a point in space falls within a occupied voxel in the volumetric map. The point is rotated, translated and quantised to voxel indices. These transformed indices are queried in the volumetric map for occupancy. Transform in this case includes the rotation, translation and quantisation. The map representations are a standard hash table, Bloom filter and dense 3D array. The dense 3D array is likely to be the fastest possible but requires an inordinate amount of memory.

world data, we seek to evaluate if the efficiency improvements observed in standalone trials (Experiment 1) translate into faster execution in real applications, and our findings indeed demonstrate this critical point.

For these experiments the Bloom filter used two hash functions and had 6 million elements. The thermolab map results in 142812 voxels at a map resolution of 0.05m.

The two hash functions applied to the voxel index are the Fowler-Noll-Vo hash function and D. J. Bernstein's hash. The pseudo code for these two hash functions follows.

```
def djbhash(i):
    h = 5381L
    t = (h * 33) & 0xffffffffL
    return t ^ i
def fnvhash(i):
    h = 2166136261
    t = (h * 16777619) & 0xffffffffL
    return t ^ i
```

A. Experiment 1: Comparative Efficiency of Lookups

Fig. 1 compares the execution times of various aspects of the process required to lookup a random point in a map consisting of randomly generated voxels. In the process of scan-to-map-matching either with global localisation or alignment with an initial pose many candidate poses are checked against the map. Each candidate pose check involves the transformation and query of a large selection of points in the scan. The point lookup procedure is therefore executed many times, and we show detailed comparative timing information for this part of the algorithm.

Query times in Fig. 1 show that map querying with Bloom tables is three times faster than with hash tables, which is the de facto method for fast lookups. A similar speed-up is shown when including the transformation steps. Also included in Fig. 1 is the time taken to calculate the overlap between the transformed scan and the map via a set intersection operation.

What is exciting about the results in Fig. 1 is how close the Bloom filter lookup time is to the fastest possible, that of a dense 3D array arranged contiguously in memory. This dense 3D array is the natural 3D extension of standard occupancy grids used in mobile robotics for many years [18]. On the experimental machine, it corresponds to nearly 10 million points queried against the map per second.

B. Experiment 2: Simulation with Perfect Ground Truth

We now test our system using simulated data for which perfect ground truth is available. The operant hypothesis is that the accuracy of our system results will not degrade from the use of Bloom filters for lookup, which is a potential source of error. The simulated data for these experiments was generated by building an indoor scene in Blender and having the camera follow a path through the scene. At a number of points along the path the view from the camera is rendered as a depth image. These depth images (Fig. 2) are converted to point clouds and fed to the multi-resolution alignment process described in Section III. No noise is added artificially but inaccuracies in the resulting point clouds do arise from the low colour depth (one byte) of the depth images. The resulting map along with the trajectory of camera positions is displayed in Fig. 3.

With the Bloom filter optimization, our position errors are 0.02m and orientation errors are 0.3 degrees for a map resolution of 0.04m. Good mapping accuracy corresponds to a low growth in the map occupied voxel count because if a new scan is aligned well to the map more of the new scan points should fall in voxels which are already occupied in the map thus the number of new occupied voxels needed in the map should be lower. This is a useful map quality metric when no pose ground truth is available. Although the Bloom filter is probabilistic Fig. 5 highlights there is no reduction in mapping accuracy as indicated by the map occupied voxel counts. The lines for the hash table and Bloom performance are coincident and as to be expected are significantly better than mapping with odometry alone. To check the effect of the Bloom filter approximation experiments were run with and without Bloom filter acceleration. With a Bloom filter size of 6×10^6 the pose solutions obtained were identical however at 6×10^5 a position error of around 0.02m was introduced.

C. Experiment 3: Benchmark thermolab Data

In order to confirm that the improvements observed in Fig. 1 translate into faster execution for real applications we consider the example application of SLAM. SLAM by scan to map alignment was performed on the benchmark *thermolab* dataset using Bloom filters or hash tables. The *thermolab* dataset [17] was recorded by Dorit Borrmann, and Hassan Afzal from Jacobs University Bremen gGmbH,

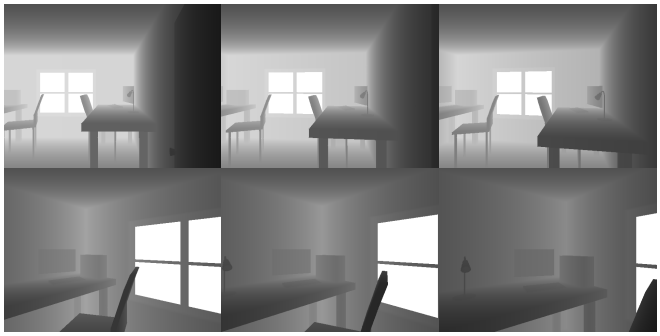

Fig. 2. Example depth images rendered for a path through a typical indoor scene by the 3D design program Blender.

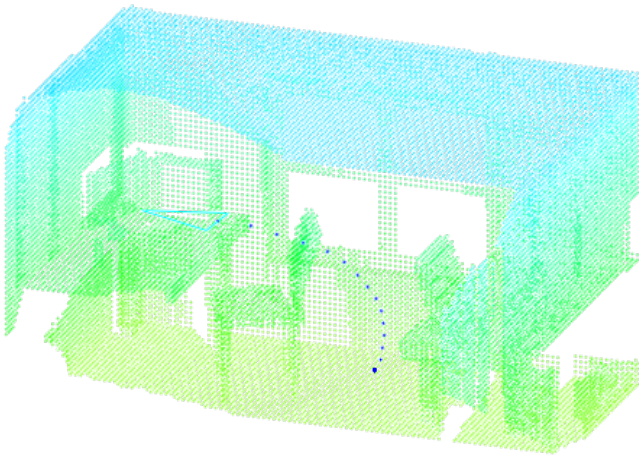

Fig. 3. Map generated from depth images in Fig. 2 the trajectory of observation points are shown in blue. The map resolution is 0.04m, the position errors are 0.02m and the orientation errors are 0.3 degrees.

Germany. The location is the Automation Lab at this university. Acquisition of range information was from a Riegl VZ-400. Each composite scan of around 400,000 points is a panoramic scan formed by rotating the scanner in place to capture 9 scans. Thermal camera information is also included in this dataset but is not used for this paper.

The resulting map generated from the *thermolab* dataset is included in Fig. 4. The execution times for each scan, for both map updating and alignment are graphed in Fig. 5. It is clear from Fig. 5 that, the improvement in query times translates into improved real application performance. Secondly, the extra Bloom filter updates required in our system for testing set membership do not significantly increase the map update times.

V. Conclusion

Our described volumetric map approach involving Bloom filters allows map lookups close to the fastest lookup possible, that of a dense 3D array, but with the memory space savings of a sparse volumetric map. The Bloom filter accelerates lookups and hence localisation and mapping with no observed impact on the mapping accuracy due to the probabilistic nature of Bloom tables (they can occasionally return false positives). The mapping accuracy was tested by

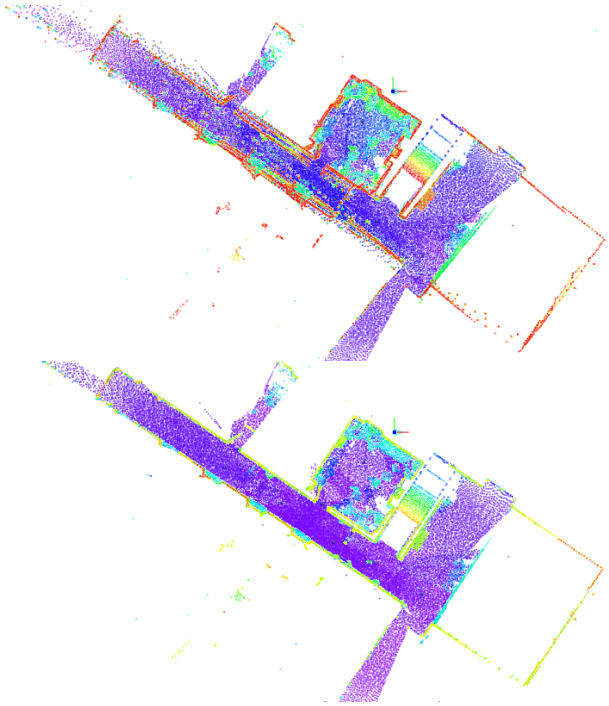

Fig. 4. Resulting volumetric map produced from the *thermolab* first by the odometry information alone provided with the dataset and second by MROL with Bloom filter accelerated alignment and map generation. The voxel size for both maps is 0.05m.

performing scan to map alignment for a simulated data set generated from a modeled 3D scene for which perfect ground truth was available and from a public benchmark data set.

For point in map queries the Bloom filter was over 3 times faster than the de facto standard hash table and perhaps more significantly within 10% of the query speed of a dense 3D array. The dense 3D is conjectured to be the fastest possible query mechanism however it cannot be used in practice due to its infeasible memory requirements. Also contributing to increased speed is a packed voxel index memory layout. In conjunction these accelerate the point-in-map collision check process threefold and enable sub-voxel alignment accuracy. Localisation accuracy for the simulated data set for voxels of 0.04m was 0.02m and 0.3 degrees. Employing a sequential multi-resolution alignment process substantially enlarges the region of convergence which is vital if initial pose estimates have large error.

In this work we test the accelerated spatial proximity data structure on the example application of SLAM with 3D scan to map matching to acquire both the poses of the scans and a good quality volumetric map.

The occasional false positives that arise from the Bloom filter might be of concern, however it is worth noting three points, firstly that this false positive error is adjustable, secondly for any practical application involving data from the real world such as mapping and path planning uncertainty is present anyway. Finally the Bloom filter can be backed by an error free data store such as a hash table and still accelerate lookups if there are a high proportion of query terms that

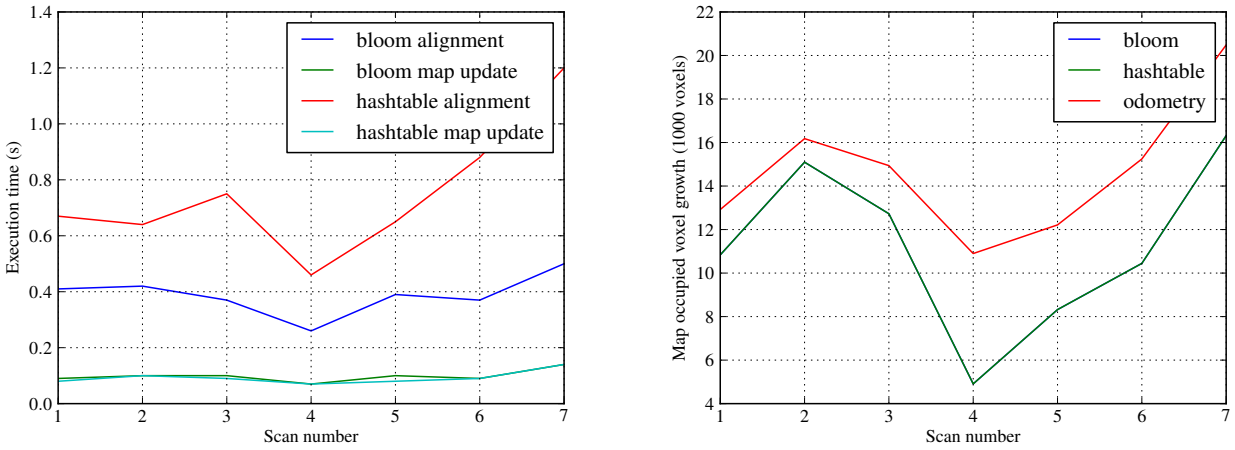

Fig. 5. Comparison between prior method involving hash tables and proposed method involving Bloom tables on the real 3D *thermolab* dataset. The first graph plots the execution time for alignment and map updating for the two methods. The second graph shows the map occupied voxel count growth of the two methods in addition to that of mapping with odometry alone. A lower growth in the map occupied voxel count is associated with better scan alignment. Note that the hash table and bloom voxel counts are identical and so the curves are coincident.

are not present in the set.

Much of the source code repository for this work, including the Bloom filter implementation, is available at [19].

ACKNOWLEDGEMENTS

This material is based upon work partially supported by the Federal Highway Administration (DTFH61-07-H-00023), the Army Research Office (W911NF-11-1-0090) and the Defense Advanced Research Projects Agency Mind's Eye Program (W911NF-10-2-0062).

REFERENCES

[1] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3D Point Cloud Based Object Maps for Household Environments," *Robotics and Autonomous Systems Journal (Special Issue on Semantic Knowledge)*, 2008. [Online]. Available: http://files.rbrusu.com/publications/Rusu08RAS-Semantic.pdf

[2] K. Pathak, A. Birk, N. Vaskevicius, M. Pfingsthorn, S. Schwertfeger, and J. Poppinga, "Online 3D SLAM by registration of large planar surface segments and closed form pose-graph relaxation," *Journal of Field Robotics: Special Issue on 3D Mapping*, vol. 27, no. 1, pp. 52–84, 2010.

[3] D. M. Cole, A. R. Harrison, and P. M. Newman, "Using naturally salient regions for SLAM with 3D laser data," in *IEEE International Conference on robotics and automation*, April 2005.

[4] P. Besl and N. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[5] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Application VISSAPP'09)*. INSTICC Press, 2009, pp. 331–340.

[6] A. Nüchter, K. Lingemann, and J. Hertzberg, "Cached k-d tree search for ICP algorithms," in *Proceedings of the 6th IEEE International Conference on Recent Advances in 3D Digital Imaging and Modeling (3DIM '07)*, August 2007, pp. 419–426.

[7] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, USA, May 2010. [Online]. Available: http://ais.informatik.uni-freiburg.de/publications/papers/wurm10octomap.pdf

[8] J. Ryde and H. Hu, "3D mapping with multi-resolution occupied voxel lists," *Autonomous Robots*, vol. 28, no. 2, pp. 169–185, February 2010.

[9] J. Ryde and N. Hillier, "Alignment and 3D scene change detection for segmentation in autonomous earth moving," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[10] J. Xie, F. Nashashibi, M. Parent, and O. G. Favrot, "A real-time robust global localization for autonomous mobile robots in large environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[11] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[12] L. Fan, P. Cao, J. Almeida, and A. Broder, "Summary cache: A scalable wide-area web cache sharing protocol," *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, pp. 281–293, 2000.

[13] O. Wulf, A. Nüchter, J. Hertzberg, and B. Wagner, "Benchmarking urban six-degree-of-freedom simultaneous localization and mapping," *Journal of Field Robotics (JFR), Wiley & Son, ISSN 1556-4959*, vol. 25 issue 3, pp. 148–163, March 2008.

[14] Q.-X. Huang and D. Anguelov, "High quality pose estimation by aligning multiple scans to a latent map," in *Robotics and Automation, 2010. ICRA '10. IEEE International Conference on*, Anchorage, Alaska USA, May 2010.

[15] J. Forsberg, U. Larsson, and Å. Wernersson, "Mobile robot navigation using the range-weighted Hough transform," *IEEE Robotics and Automation Magazine*, vol. 2, no. 1, pp. 18–26, 1995.

[16] J. Ryde and H. Hu, "Fast circular landmark detection for cooperative localisation and mapping," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, April 2005, pp. 2756–2761.

[17] D. Borrmann and H. Afzal, "Robotic 3D scan repository," Universität Osnabrück, Tech. Rep., 2011. [Online]. Available: http://kos.informatik.uni-osnabrueck.de/3Dscans/

[18] H. Moravec, "Robust navigation by probabilistic volumetric sensing," Tech. Rep., February 2002.

[19] J. Ryde, "Multi-resolution occupied voxel lists (MROL) source code repository," https://launchpad.net/mrol.