Extracting Edge Voxels from 3D volumetric maps to reduce map size and accelerate mapping alignment

Julian Ryde and Jeffrey A. Delmerico
Computer Science and Engineering
University at Buffalo
Buffalo NY, USA
Email: {jryde,jad12}@buffalo.edu

***Abstract*—For effective mobile robots we need a concise yet adequately descriptive mechanism for representing their surroundings. Traditionally 2D occupancy grids have proven effective for task such as SLAM, path planning and obstacle avoidance.**

Applying this to 3D maps requires consideration due to the large memory requirements of the resulting dense arrays. Approaches to address this, such as octrees and occupied voxel lists, take advantage of the relative sparsity of occupied voxels. We enhance the occupied voxel list representation by filtering out those voxels that are on planar sections of the environment to leave edge-like voxels. To do this we apply a structure tensor operation to the voxel map followed by a classification of the eigenvalues to remove voxels that are part of flat regions such as floors, walls and ceilings. This leaves the voxels tracing the edges of the environment producing a wire-frame like model. Fewer edge voxels require less memory and enable faster alignment.

We compare the performance of scan-to-map matching of extracted edge voxels with that of the corresponding full 3D scans. We show that alignment accuracy is preserved when using edge voxels, while achieving a five times speedup and reduced memory requirements, compared to matching with all occupied voxels. It is posited that these edge voxel maps could also be useful for appearance based localisation.

***Index Terms*—voxels; edge detection; mapping; localisation; SLAM**

I. Introduction

Indoor environments are typically full of planar surfaces (floors, walls, ceilings, furniture, etc.), and consequently numerous researchers have explored plane-based mapping [1]. There are limitations to this assumption, where a plane-based approach is not appropriate: natural outdoor environments, building with curved surfaces, and cluttered scenes, for example those found in search and rescue scenarios. Although the use of planes as landmarks can be a good way of compressing map information, feature points and edges are more effective at constraining a robot's pose than an observed planar surface. We propose an extension to voxel based mapping which accomplishes mapping by aligning edges voxels extracted from a 3D scan or range image to an edge voxel map.

An exciting aspect of edge voxel maps is that we can localise range and image sensors with the same map.

For many structured environments, considering only edge voxels removes the dominant planar surfaces in the scene: floors, walls and ceilings. The remaining voxels are those that lie along the intersection of planes or in cluttered regions.

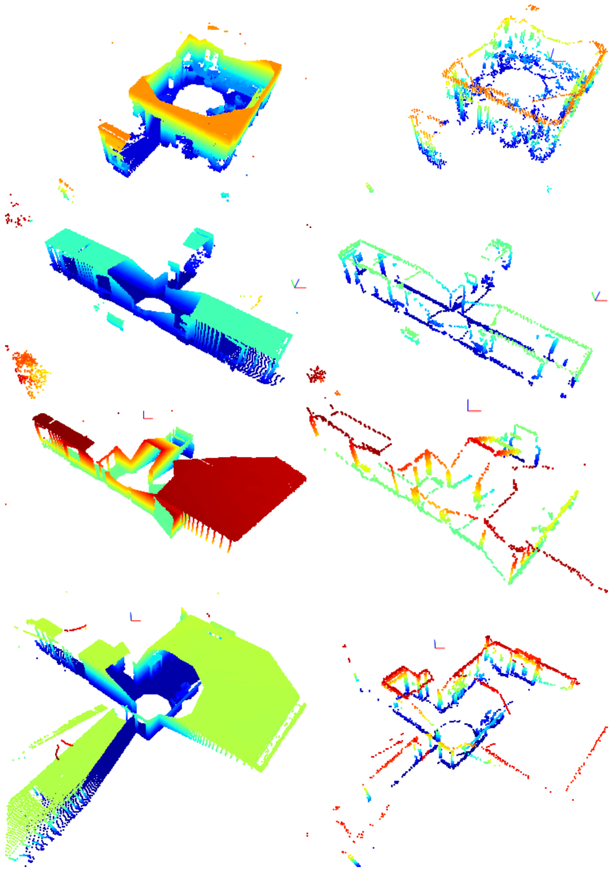


Fig. 1. Examples of scans from the *thermolab* dataset (left) and the corresponding edge voxels extracted from them (right).

Whilst feature-based SLAM has proven effective in many situations, it relies on successful feature extraction and identification that can not be guaranteed. Scan-to-map matching approaches do not require feature extraction or association and therefore are immune to this problem. However, they require much more memory and computation to perform. Our proposed approach avoids both of these problems by reducing the memory and computational load of scan-to-map matching while leveraging pose-invariant structures in the environment.

II. Background material/Related work

The edges that manifest in images are regions of high contrast that usually follow a path in the image. These edges are due to four main reasons.

- Appearance
  - Texture change — Abrupt change in surface colour
  - Lighting change — Sharp shadows
- Geometry
  - Range discontinuity — Abrupt change in distance from the observer
  - Surface normal change — E.g. intersection of two planes

It is important to appreciate the distinction in the causes of image edges. Texture change and illumination edges are not observed by 3D sensors. So the remaining geometric edge types are range discontinuities and abrupt surface normal changes. Surface normal changes are pose invariant, however edges due to range discontinuities are not. For instance, consider a cylinder and the range discontinuity edge of the curved source of the cylinder. The position of this edge varies in 3D space as the position of the observer shifts.

In the same way that edges in images require a fairly flexible definition, a rigorous definition of edge voxels is somewhat elusive. For instance a definition could be those voxels that do not contain points that lie in a plane, however this would classify voxels on a curved surface (such as a cylinder) as edges. A more robust definition might describe edge voxels as lying on a smooth manifold in 3D space, but the quantised nature of a voxel grid makes this description flawed. On account of this, we assess our edge extraction techniques on a fit-for-purpose basis, rather than by comparison to ground truth, which may be subjective.

Sparse maps can be made that contain only positions of landmarks or features that aid in localisation, but denser maps consisting of point clouds or occupancy grids can also enable obstacle avoidance and path planning. Denser maps, however, require more memory to store and are more computationally expensive to build.

The following list summarises these maps in descending sparsity and places the edge voxel maps into context of the broader research. Sparser maps are smaller, easier to store and quicker to process however they do not work with as wide a range of robot tasks.

- 3D occupancy grid — Extension of 2D occupancy grids
- Occupied voxel list — Occupied voxels only
- Edge voxel map — Non planar voxels
- Feature map — List of point features and their covariances

Edges have been previously incorporated into feature maps, but no work to our knowledge approaches edges in 3D mapping as a subset of the occupancy grid. In [2] they extend conventional landmark based SLAM to incorporate edge information by the extraction of edgelets from the scene image. However, the resulting map is still a subset of the edge pixels visible in the scene. The Point Cloud Library (PCL) [3] also includes a great deal of functionality for working with point clouds, and extracting planes and feature points. However, their approach is decidedly tied to a point cloud representation, and they do not incorporate functionality for extracting edges from them.

III. Methods

A. Edge Voxel SLAM

Many contemporary SLAM approaches rely on the successful and reliable extraction of point features or identifiable landmarks. This reliance on feature detection can be problematic in certain environments lacking such distinctive landmarks. Especially in indoor environments whist distinctive points are sometimes not observable there are usually geometric edge-like regions such as at the intersection of the walls and floor.

As a last resort full matching of the occupied voxels is always possible if insufficient edge voxels are detected. Generating maps of the geometric edge voxels is faster and requires fewer voxels to express the map at a given resolution, consequently edge voxel maps require less memory.

The SLAM approach we use to test the efficacy of edge voxels in a concrete application is based on multi-resolution occupied voxel lists detailed in [4]. We employ an iterative voxel based alignment strategy where the scan is first aligned to the coarsest map resolution. The scan is sampled and then transformed by the initial guess pose and the number of points falling inside map voxels counted. The initial pose can be provided by either odometry or the pose of the previous scan if they are close enough. Nearby surrounding poses are also tested and current best pose updated to that with the highest coincident points in the map. The search proceeds until it converges on a (quite possibly local) maximum overlap. The procedure is then performed on the next finest resolution with voxels of half the width starting with the best pose from the larger resolution and the best pose found so far adjust accordingly. This is repeated at all the resolutions stored, which is typically three to five. For more information regarding this multi-resolution alignment process see [5].

This procedure can be made faster by reducing the number of points that need to be considered for each scan, hence the initial sampling step. By calculating the points that lie in edge voxels and aligning those to an edge voxel map the alignment procedure is significantly accelerated whilst requiring less memory for the storage of the map.

An edge voxel classification algorithm has to be applicable to both a single 3D range scan as well as a map consisting of the result of a number of merged 3D scans. Performing edge voxel extraction on a full map produces better results due to the greater accuracy and density of the occupied voxels. The volumetric edge extraction is more accurate when there is greater information on the occupancy state of surrounding voxels.

A single 3D scan is limited in range, and will thus contain points that represent the manifestation of this limitation, and not actual object boundaries in the environment. The different kinds of border points in a scan are enumerated by [6]: obstacle border points, veil points and shadow border points. A single scan will have more spurious edge voxels due to occlusion, boundary shadows and data sparsity at long range. A full map fuses points from many scans together and can mitigate the latter two categories, which are viewpoint dependent. This leaves only legitimate view-invariant object border points. However, these spurious voxels are often categorised as edges in structure tensor analysis, and in the sparser edge-extracted scans, their presence may complicate the scan alignment process.

There are two points at which to perform edge voxel extraction, either on a single scan like [6] or on the entire map. In this work we demonstrate good performance on a scan by scan basis, which is necessary for real-time operation. Unfortunately, results in artifact edge voxels arising from occlusions and sensor noise and a slightly poorer quality edge map. We anticipate that the edge voxel extraction can be rerun on a combined map of all occupied voxels to produce a better quality edge map if required. We process the scans individually and achieve comparable alignment accuracy to using the full scans, but our approach to edge extraction can be reapplied to the full map or iteratively during map building in order to remove the spurious view-dependent edge voxels.

B. Edge Voxel Extraction

Extracting the edge-like voxels from 3D data can be performed on either the original point cloud or its voxel representation. Since a 3D occupancy grid is a 3D structural analogue of a 2D image, approaching this problem using a voxel representation permits much of the work from the computer vision community to be extended to 3D and applied.

Finding edges in 3D is analogous to finding corners in 2D images. In image processing, a corner is a 0D entity in the 2D image, and for volumetric edge extraction we are looking for 1D edges throughout a 3D volume. In both cases, this is a reduction of 2 dimensions. If we consider feature extraction a method of dimension reduction, then these approaches are equivalent. By this analysis, determining 1D edges in a 2D image is equivalent to extracting 2D planes in 3D, both a dimension reduction of 1D.

On the other hand, if we operate directly on the point cloud, then algorithms based on Principal Component Analysis (PCA) can be applied. Groups of points (either within a voxel, or a point's k nearest neighbors) can be analysed and the eigenvalues of the resulting covariance matrix can be used to determine the planar nature of each point. Points corresponding to planar regions can be removed from each scan prior to map matching. For many indoor environments, in principle this should remove a large number of the occupied voxels leaving a subset that include geometric edge voxels.

The main advantage of point based metrics is that they can operate at a higher resolution than occupancy grid approaches. However, the number of points recorded in the map grows unbounded with time. This map growth occurs for a continuously operating mobile robot even if it does not explore new areas, making purely point-based approaches unscalable.

Secondly, a voxel grid can encode the difference between unknown and free space. This information can improve edge classification in the face of incomplete data. For instance, apparent edge voxels bordering on unknown regions should not be classified as edge voxels; it is possible that the neighboring unknown voxels are actually occupied and are as yet unobserved.

Sources of inspiration for volumetric edge detection come from the computer image processing literature including Haar features [7] and bilateral filtering which is mentioned in [6] and used in [8]. Experiments with many edge extraction methods (PCA, Difference of Gaussians, template matching, Haar filter responses) did not produce edge voxels corresponding to our intuition about them, did not remove planar voxels well due to noisy point data, or were limited to axis aligned edges. Instead, we employ an approach based on eigenvalue analysis of the structure tensor, computed for occupied voxels in a 3D occupancy grid.

Structure Tensor Analysis

Similar to how PCA can be applied to the original points we calculate the discrete structure tensor on the 3D volumetric occupancy grids. The structure tensor describes the size and direction of the gradients in scalar field surrounding a point in space. This gradient information is encapsulated in the eigenvalues and corresponding eigenvectors of this structure tensor. A large eigenvalue indicates that there is a large local gradient in the direction of its corresponding eigenvector. We consider only the eigenvalues to determine whether an occupied voxel is edge-like.

The Harris corner detection method [9] leverages the 2D structure tensor to extract edges and corners in images. Analysis of the 2×2 structure tensor for an image pixel can allow its classification as an edge pixel (one large and one small eigenvalue), a corner (two large eigenvalues), or a pixel that is not of interest (two small eigenvalues).

Similarly, in three dimensions the relative magnitudes of the structure tensor eigenvalues permit classification of voxels. A planar voxel will have one direction with a large gradient (normal to the plane) and two directions with small gradients (orthogonal directions in the plane), and will thus have one large and two small eigenvalues. A line or edge in three dimensions will be characterized by two directions of large gradient, and one direction with small gradient (along the edge). Thus, an edge voxel will have two large and one small eigenvalues. An isolated region in space will have large gradients in every direction, and so all of its structure tensor eigenvalues will be large. Similarly for a homogeneous region, the gradient will be small in every direction, so all of its eigenvalues will be small.

There does not appear to be any previous attempts at using structure tensor analysis for edge detection on voxelized data. The nearest is the applying the 3D structure tensor to video data [10], [11].

Structure Tensor Derivation

The 3D structure tensor is derived from the weighted sum of squared differences between shifted volume patches. Consider a subvolume V of an occupancy grid I and the same volume shifted by (x, y, z). The weighted sum of squared differences for that shifted patch, $S(x, y, z)$, is:

$$S(x,y,z) = \sum_{v \in V} w(v)(I(v + (x,y,z)) - I(v))^2 \tag{1}$$

where w is some weighting function defined over each voxel v in the subvolume. Using the Taylor series expansion of $I(v + (x, y, z))$:

$$I(v + (x,y,z)) \approx I(v) + I_x(v)x + I_y(v)y + I_z(v)z \tag{2}$$

this can be simplified to:

$$S(x,y,z) = \sum_{v \in V} w(v)(I_x(v)x + I_y(v)y + I_z(v)z)^2 \tag{3}$$

which can be written in matrix form as:

$$S(x,y,z) \approx \begin{pmatrix} x & y & z \end{pmatrix} A \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{4}$$

where A is the structure tensor for the original volume:

$$A = \sum_{v \in V} w(v) \begin{bmatrix} I_x^2 & I_xI_y & I_xI_z \\ I_xI_y & I_y^2 & I_yI_z \\ I_xI_z & I_yI_z & I_z^2 \end{bmatrix} \tag{5}$$

Computed over a subvolume centered at some voxel $p = (x_p, y_p, z_p)$ in the occupancy grid, the eigenvalues of the structure tensor will summarize the gradient structure in that local neighborhood around p, and can be used to determine if p is an edge or not.

In order to compute the structure tensor eigenvalues for each voxel in an occupancy grid, we proceed in the following steps. The occupancy grid (which is binary) is first pre-smoothed with a multivariate Gaussian filter, which additionally permits easy computation of the partial derivatives with Gaussian partials instead of finite difference estimates. After smoothing with a kernel having half-width h, the partial x derivative at a voxel $p = (x_p, y_p, z_p)$ can be computed with:

$$I_x(p) = -x \exp\left[-\left(\frac{x_p^2}{(\frac{h}{2})^2} + \frac{y_p^2}{(\frac{h}{2})^2} + \frac{z_p^2}{(\frac{h}{2})^2}\right)\right] \tag{6}$$

with the y and z partial derivatives computed similarly. Note, we ignore the scale factor of the Gaussian derivatives as it results in a constant factor on the structure tensor that scales all of the eigenvalues at all voxels equally. Therefore, we omit this extra computation and incorporate the constant factor into the threshold for edge voxels.

Then for each occupied voxel p, we consider a $k \times k \times k$ neighborhood centered at p (a neighborhood of half-width h such that $k = 2h + 1$). For each voxel in that neighborhood, we use a 3D Gaussian weighting function:

$$w = \exp\left[-\left(\frac{x_p^2}{h} + \frac{y_p^2}{h} + \frac{z_p^2}{h}\right)\right] \tag{7}$$

and as with the Gaussian derivatives, we omit the scale factor. This is combined with the previously computed partial derivatives to determine the entries of the structure tensor at p.

Eigenvalue Classification

Our approach to selecting edge voxels based on their eigenvalues is motivated by the physical meaning of the structure tensor and reinforced by an analysis of real world scan data. Much like the eigenvalues and eigenvectors of the covariance matrix in Principal Component Analysis explain the orthogonal directions of greatest variance in point data, the eigenvalues and eigenvectors of the structure tensor reveal the directions of greatest gradient in quantised data. Geometrically, we expect that edge-like structures in the point cloud data will have two orthogonal directions with large gradients and one direction along the edge with a small gradient. Thus we seek the corresponding voxels that have two large structure tensor eigenvalues and one small one. We enforce this property by thresholding the eigenvalues for both their magnitudes as well as the ratio of the middle and largest eigenvalues.

Our motivation for thresholding comes from an analysis of real world data from the *thermolab* dataset as shown in Fig. 2. From the set of all occupied voxels, we select candidate edge voxels that have their smallest structure tensor eigenvalue < 0.1 and their middle and largest eigenvalues > 0.1. We selected this threshold value empirically based on the histograms; we can see that most voxels pass this test for the smallest and largest eigenvalues, but it provides us with some discriminating power at the middle eigenvalue. This threshold excludes the lowest peak of the distribution of middle eigenvalues, allowing us to only consider voxel candidates that have two reasonably large eigenvalues.

The other geometric factor affecting our analysis is the need for edge voxels to have two large eigenvalues of approximately equal magnitude. Since a voxel in a planar region could have a middle eigenvalue > 0.1, but a largest eigenvalue that is dramatically larger, we compute the ratio of the middle to largest eigenvalues and threshold that, selecting as edges only those voxels with a ratio > 0.2. We determined this threshold value empirically by considering the distribution of the middle and smallest eigenvalues normalized by the largest (i.e. middle/largest vs. smallest/largest) as seen in Fig. 3. In this figure, the vertical axis represents the ratio we are thresholding, and the color scale goes from zero frequency in navy blue to the highest frequency in red. The bottom left corner represents planar voxels whose smallest and middle eigenvalues are very small relative to the largest. The upper left corner represents ideal edges where the middle and largest eigenvalues are equal and both large relative to the smallest. The top right corner represents corners, isolated points, or homogeneous regions where all three eigenvalues are approximately the same size. Since our three eigenvalues are sorted, we do not expect any in the lower right triangle of the plot. A threshold on the middle/largest ratio is equivalent to selecting the voxels that fall into bins above a particular horizontal line.

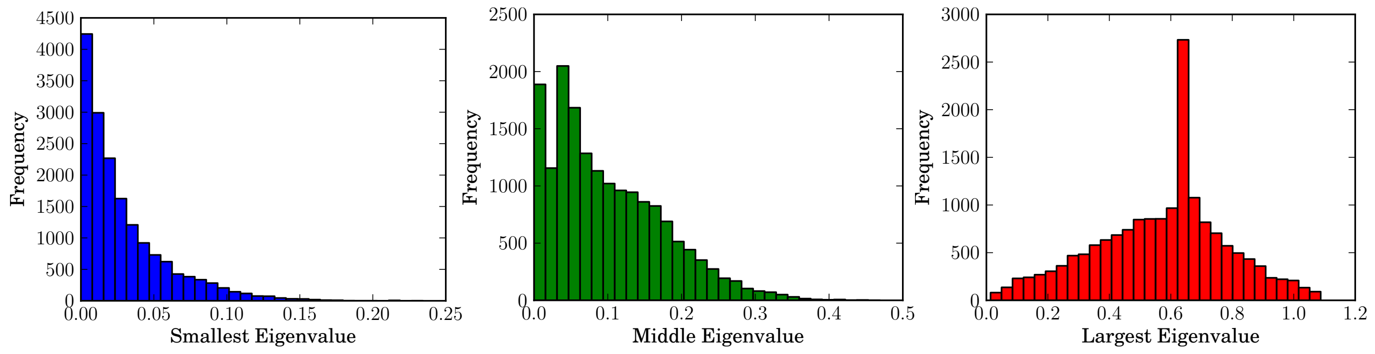


Fig. 2. Histogram plots of the smallest (first), middle (center), and largest (last) eigenvalues taken for over the first scan in the *thermolab* dataset.

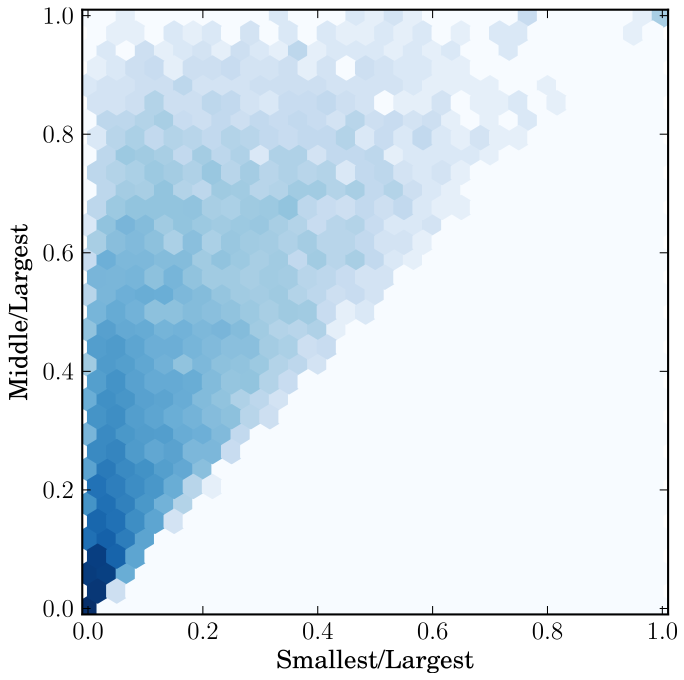


Fig. 3. 2D Histogram plot of middle and smallest eigenvalues normalized by largest eigenvalue for each voxel in the first scan of the *thermolab* dataset. Ideal edges (two large, equal eigenvalues) are in the upper left corner, planes (one large, two small eigenvalues) are in the bottom left corner, and corners or isolated/homogeneous regions (three equal eigenvalues) are in the upper right corner. Here, blue represents the lowest frequency and red represents the highest frequency (view in color).

By placing the threshold at 0.2, we remove the planar voxels in the bottom left corner of Fig. 3. Above this threshold, however, we retain not only the ideal edges and corners, but the other edge voxels that in real scan data are subject to noise and sensor limitations and do not have ideal eigenvalues. Modulating the ratio threshold permits different amounts of these non-ideal edges to be extracted, resulting in a continuum of progressively more dense edge scans. Thus the value of this threshold modulates a tradeoff between scan compression and the matching accuracy that comes from a fuller scan. From our experiments, we found that accurate mapping could be performed across nearly all of this continuum of edge voxel densities.

We require candidate voxels to pass both the eigenvalue magnitude and eigenvalue ratio tests in order to be extracted as an edge voxel. Several examples of scans from the *thermolab* dataset are shown in Fig. 1 along with their extracted edge voxels.

The previous structure tensor approaches to video segmentation [10], [11] also compute some metrics on the eigenvalues, but focus more on coherency measures. Wang and Ma [11] do compute an edge measure

$$\left(\frac{largest - middle}{largest + middle}\right)^2$$

that also seeks to capture the relationship of the largest two eigenvalues, but they provide no motivation for that particular metric, and instead we chose our system based on the above analysis of our data, which is very different from the dense data of video.

Practically in our experiments edge voxels are extracted from points quantised to a cubic lattice with a given cell size and the number of points associated with each cell stored in a dense array.

The structure tensor eigenvalues are calculated for each occupied voxel and the edge voxels identified by thresholding the eigenvalues, as described above. This procedure is summarized in Algorithm III-B.

IV. Edge Voxel Extraction and Mapping Experiments

We test the effectiveness of edge voxel extraction preprocessing for scan-to-map matching. We perform multi-resolution occupied voxel list mapping as in [5]. We evaluate performance based on speed, memory usage, and alignment accuracy.

We evaluate speed by timing how long it takes to find the best transform which aligns a scan to the map and how long to update the map given the scan and the alignment transformation. These timings are performed on a single 3GHz CPU. We have not included graphs of the timing information for the edge voxel extraction because our current implementation for this is inefficient operating on dense arrays rather than taking advantage of the sparsity of occupied voxels. The edge voxel

Algorithm 1 Edge Voxel Extraction

```
Load point cloud from sensor into occupied voxel list $L$ of resolution $r$.
Let $V$ be the set of all occupied voxels in $L$
Let $N$ be the set of all voxels within a $k \times k \times k$ neighborhood of a voxel in $V$
Let $S = V \cup N$
for all $s \in S$ do
    Compute $I_x(s)$, $I_y(s)$, and $I_z(s)$
    Compute $(I_x(s))^2$, $(I_y(s))^2$, $(I_z(s))^2$, $I_x(s)I_y(s)$, $I_x(s)I_z(s)$, and $I_y(s)I_z(s)$
end for
Let $G$ be the $k \times k \times k$ Gaussian kernel
Let $E$ be an empty occupied voxel list to hold edge voxels
for all $v \in V$ do
    Compute $A_{xx} = G * (I_x(s))^2$ (similarly for $A_{xy}$, $A_{xz}$, $A_{yy}$, $A_{yz}$, and $A_{zz}$)
    Compute the structure tensor for $v$:
        $A(v) = \begin{bmatrix} A_{xx} & A_{xy} & A_{xz} \\ A_{xy} & A_{yy} & A_{yz} \\ A_{xz} & A_{yz} & A_{zz} \end{bmatrix}$
    Determine the eigenvalues for $A(v)$ and sort them in descending order: $\lambda_1 \geq \lambda_2 \geq \lambda_3$
    Classify $v$ based on the eigenvalues and add $v$ to $E$ if $v$ is an edge voxel
end for
```

extraction operation currently takes around 10 seconds for a single scan.

The memory requirements are dominated by storage for the map. It consists of list-based representations of either all of the occupied voxels or only the extracted edge voxels. To illustrate the memory requirements, Fig. 7 and Fig. 6 include the total voxels in the map as subsequent scans are aligned.

The experiments were performed with two datasets, *blender scene* and *thermolab*. The first dataset, *blender scene*, consists of depth images; Fig. 4 generated by a depth map rendering of an indoor a scene hand modeled in Blender, a 3D animation program.

The second dataset, *thermolab*, was recorded by Dorit Borrmann, and Hassan Afzal from Jacobs University Bremen gGmbH, Germany, [12]. The location is the Automation Lab at this university. Acquisition of range information was from a Riegl VZ-400. Each of the 9 composite scans contains around 400,000 points in a panorama formed by rotating the scanner in place. Thermal camera information is also included in this dataset but is not used for this paper.

Although we have refrained from testing characteristics of edge voxel extraction believed to be beneficial, it is worth discussing them in the light of the results. Desirable properties of extracted edge voxels include the following: they should be generally invariant to rotation and translation, they should be helpful in terms of constraining robot pose, and they should be an open enough class that some edge voxels can be extracted

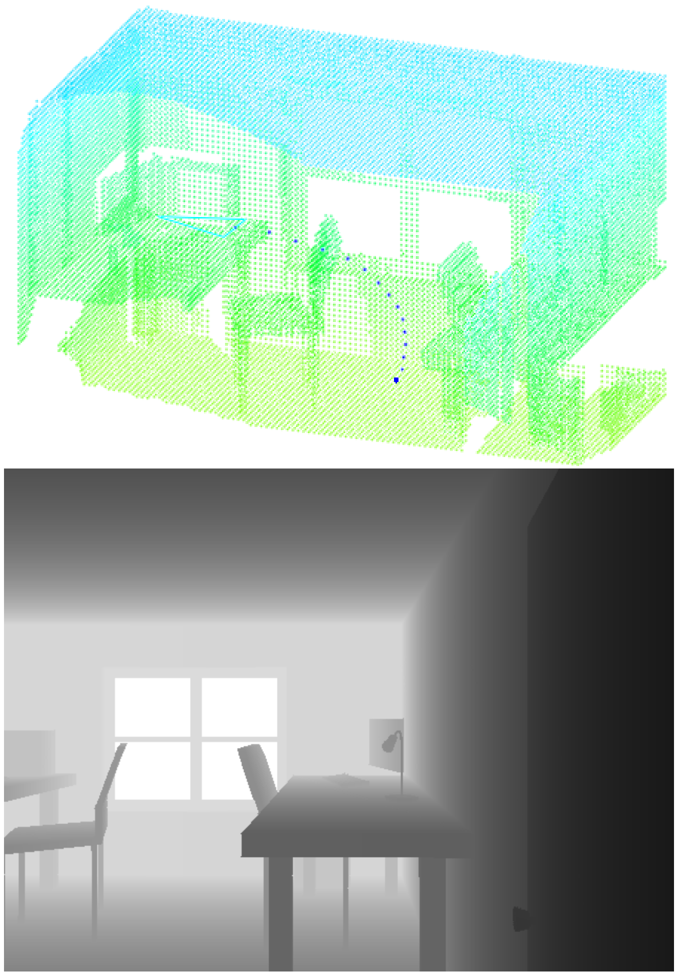


Fig. 4. Example depth image and resulting occupied voxel map.

in the vast majority of situations. The extraction method must be fast to compute and must eliminate a high percentage of voxels found in typical mobile robot data, while ensuring that the aforementioned properties are satisfied.

Figure 6 shows that mapping can be performed with comparable accuracy to both odometry-based ground truth and full-scan alignment. We judge accuracy here by map voxel count, on the intuition that misaligned scans will result in more occupied voxels in the finished map. In this case, we performed full scan mapping, edge voxel mapping, and mapping using the ground truth poses. The poses determined during edge voxel mapping were then used as the fixed poses in mapping using the full scans. Although edge voxel mapping provides somewhat less of a refinement to the odometry poses than full scan mapping, the result is still a significant improvement over the odometry and is comparable to that of the full scan approach.

Our proposed method is robust and effective in extracting edge voxels that exhibit the characteristics that we desire, and the results of mapping experiments indicate that edge voxels perform as well as full scans in mapping accuracy but require significantly less memory and computational expense.

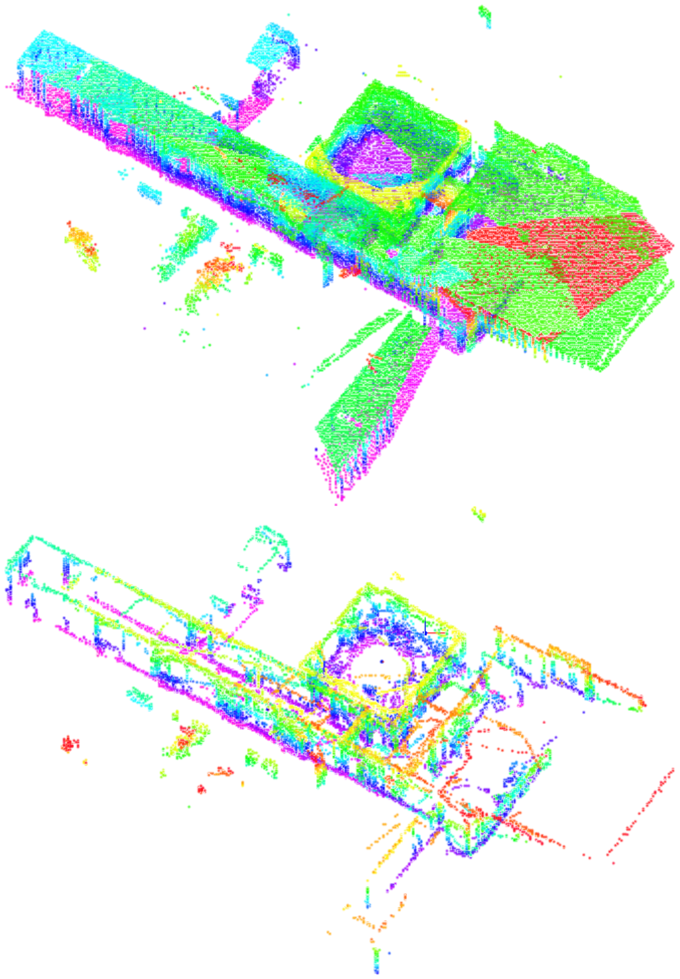
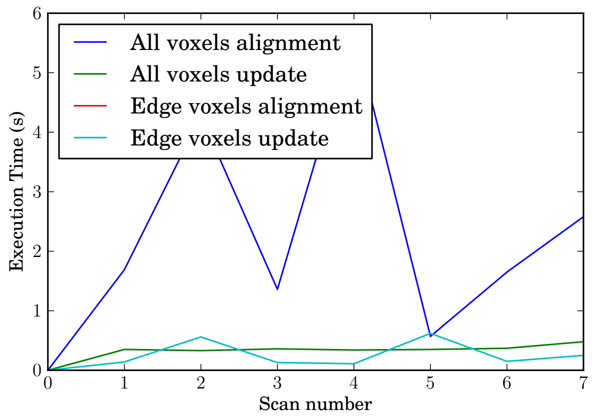


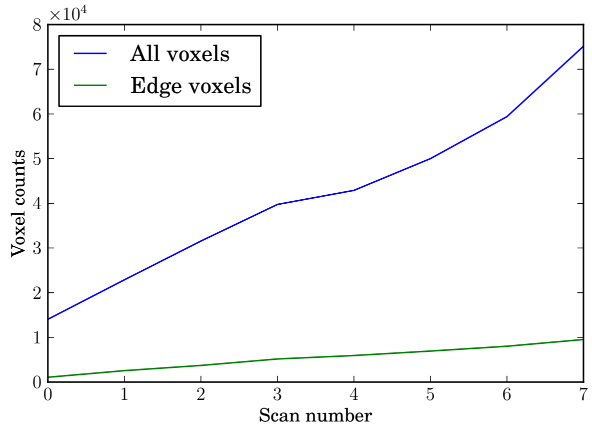
Fig. 5. The map produced using full scans from the *thermolab* dataset (top), and the map produced using extracted edge voxels (bottom).

Our edge extraction method is not yet optimized for speed and scalability, but we are currently developing a sparse representation that will permit fast extraction of edges as well as refinement of the final map with edge extraction post-processing.

V. Conclusion

We apply a novel voxel edge classification based on the 3D structure tensor to specifically excluded planar regions and perform scan-to-map matching of the resulting wire-frame like models. Whereas others have performed plane or straight line fitting on point clouds and matched those features when building a map, we operate on the voxels rather than points, and remove those voxels in planar regions. The edge voxels arising from the rejection of planar regions can lie on curved lines and our approach works in cluttered environments. Rejection of planar voxels is motivated by the planar structure of man-made environments. The majority of points in scans of such spaces will be on flat walls, floors, and ceilings, and thus their removal results in significant compression of the scan and corresponding map data. Their more frequent occurrence





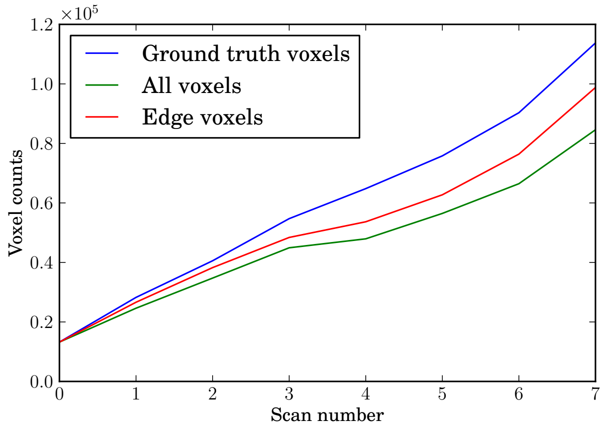


Fig. 6. Execution timings, map memory requirement, and alignment accuracy for the *thermolab* dataset. Alignment accuracy is shown as voxel counts for maps produced by full scan alignment, full scan alignment with poses determined by edge voxel alignment, and by ground truth poses.

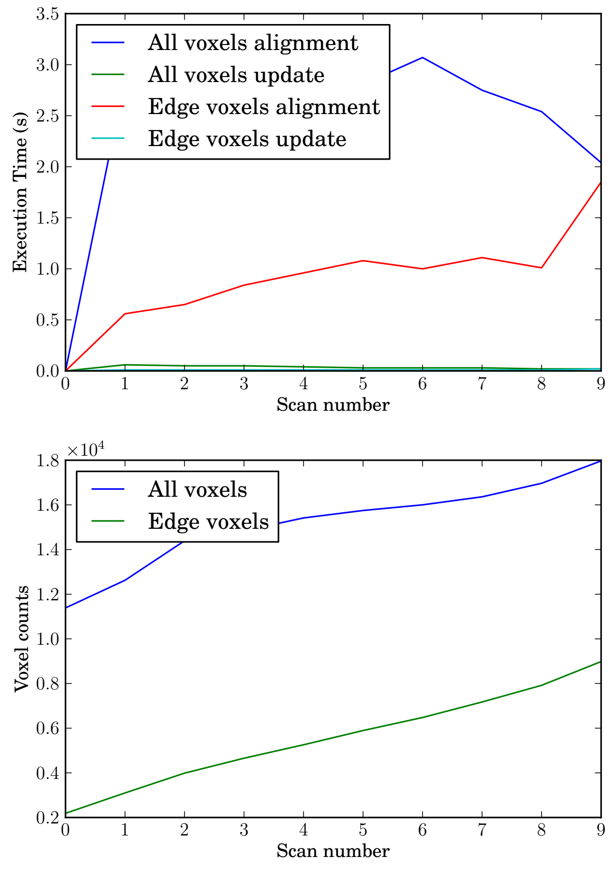


Fig. 7. Execution timings and map memory requirement for simulated depth images.

in structured environments makes them less distinctive and consequently less useful for scan-to-map matching and pose determination.

Currently, the edge extraction operation is conducted on an occupancy grid representation of the data. Scan data, however, is quite sparse, so this is not an efficient or scalable implementation. We are developing an implementation of this approach that leverages the occupied voxel lists of the MROL library to perform the edge extraction from a sparse representation of the data. This should increase execution speed and reduce memory requirements.

We aim to improve the quality of edge voxel extraction and resolve the aformentioned problem of spurious boundary voxels by accounting for the difference in unknown and free voxels. In our current implementation, only occupied voxels are maintained, and there is no distinction between free space that has been observed and unobserved, unknown voxels. Consequently, many of the border pixels in each scan, which have many unknown voxels in their neighborhoods, are erroneously classified as edge voxels. Over time this results in the degradation of the map quality as erroneous edge voxels accumulate in the map. We plan to distinguish unknown and free-space voxels in the future, which should improve map quality and scan alignment accuracy.

The extraction of the edge voxels by filtering out the often numerous planar voxels dramatically reduces the map size and accelerates the alignment. In the experiments presented, involving depth images from a model scene and real laser range data, we record a five fold decrease in map storage and a similar increase in alignment speed (Fig. 7 and Fig. 6). Even if edge voxel extraction fails, this approach can still seamlessly fall back to full occupied voxel matching making its operation robust in many environments. Finally, these edge voxel maps could be used to localise both range sensors, as demonstrated here, and in the future cameras with suitable image edge extraction.

Acknowledgements

This material is based upon work partially supported by the Federal Highway Administration under Cooperative Agreement No. DTFH61-07-H-00023 and the Army Research Office (W911NF-11-1-0090). Any opinions, findings, conclusions or recommendations are those of the authors and do not necessarily reflect the views of the FHWA or ARO.

References

[1] 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.

[2] E. Eade and T. Drummond, "Edge landmarks in monocular slam," in *In Proc. British Machine Vision Conf*, 2006.

[3] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

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

[5] 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.

[6] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard, "Point feature extraction on 3D range scans taking into account object boundaries," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[7] X. Cui, Y. Liu, S. Shan, X. Chen, and W. Gao, "3D haar-like features for pedestrian detection," in *Multimedia and Expo, 2007 IEEE International Conference on*, July 2007, pp. 1263–1266.

[8] A. Ansar, A. Castano, and L. Matthies, "Enhanced real-time stereo using bilateral filtering," in *2nd Int. Symp. on 3D Data Processing*, 2004.

[9] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey vision conference*, vol. 15. Manchester, UK, 1988, p. 50.

[10] G. Kuhne, J. Weickert, O. Schuster, and S. Richter, "A tensor-driven active contour model for moving object segmentation," in *Proceedings of the International Conference on Image Processing*, vol. 2. IEEE, 2001, pp. 73–76.

[11] H. Wang and K. Ma, "Automatic video object segmentation via 3d structure tensor," in *Proceedings of the International Conference on Image Processing*, vol. 1. IEEE, 2003, pp. I–153.

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