Proceedings of Australasian Conference on Robotics and Automation, 2-4 Dec 2013, University of New South Wales, Sydney Australia

Advantages of Exploiting Projection Structure for Segmenting Dense 3D Point Clouds

Alex Bewley and Ben Upcroft
Queensland University of Technology, Brisbane, Australia
{aj.bewley,ben.upcroft}@qut.edu.au

Abstract

Timely and comprehensive scene segmentation is often a critical step for many high level mobile robotic tasks. This paper examines a projected area based neighbourhood lookup approach with the motivation towards faster unsupervised segmentation of dense 3D point clouds. The proposed algorithm exploits the projection geometry of a depth camera to find nearest neighbours which is time independent of the input data size. Points near depth discontinuations are also detected to reinforce object boundaries in the clustering process. The search method presented is evaluated using both indoor and outdoor dense depth images and demonstrates significant improvements in speed and precision compared to the commonly used Fast library for approximate nearest neighbour (FLANN) [Muja and Lowe, 2009].

1 Introduction

Modern sensors such as time-of-flight cameras, Microsoft Kinect, calibrated stereo cameras and high definition 3D LIDAR provide rich depth information with rates of over a million points per second. Timely and comprehensive scene understanding through interpretation of this data is critical to effective decision making in mobile robotic applications. A common approach to scene understanding is to subdivide sensor data into smaller meaningful portions which can later be classified if the target application requires. While segmentation is ubiquitous in the computer vision and robotics communities, many approaches assume specific domain knowledge in the form of model fitting or supervised learning. Unsupervised methods such as spatial clustering utilise nearest neighbour techniques to efficiently partition similar data into groups which differ from other groups.

One of the biggest drawbacks of using spatial clustering in robotics has been the excessive computational load primarily due to nearest neighbour searching. While considerable attention has been given to improving the computational efficiency of nearest neighbour problems [Elseberg *et al.*, 2012], the overhead of preprocessing the data into an organised structure often prohibits these methods for online robotic perception problems. The primary focus of efficient nearest neighbour searching in these domains is to minimise both the query time and storage complexity of high dimensional databases at the cost of increased build time. As data is constantly acquired in robotic applications, maintaining complex data structures online induces a significant computational overhead.

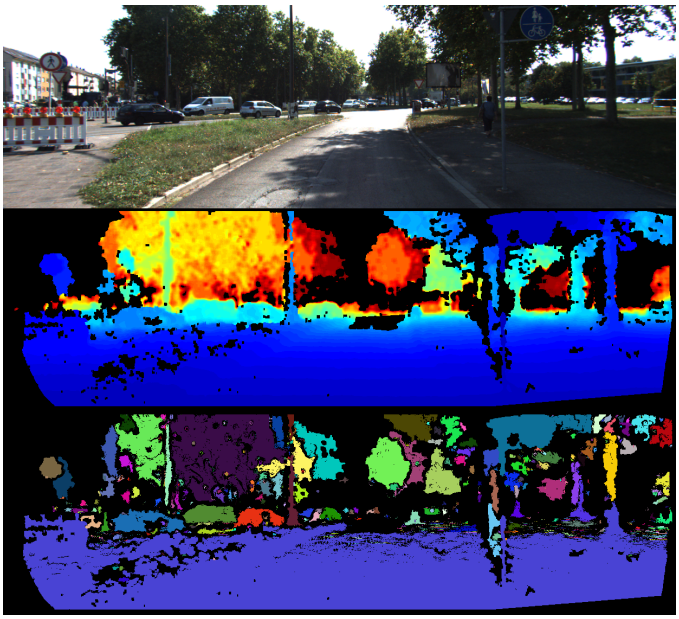


Figure 1: Top: visual frame depicting the scene for the reader's convenience. Middle: dense depth map representation of scene as input to clustering algorithm (computed using [Geiger *et al.*, 2011]). Blue represents close points and red is the maximum depth fixed at 150m. Bottom: Output of spatial clustering algorithm where each colour denotes a different cluster.

This paper takes an alternate approach for nearest neighbour searching for robotic applications where the domain is restricted to the Euclidean distance as a dissimilarity metric in $\mathbb{R}^3$ space. We exploit the 2.5D nature of dense depth maps by utilising the projection properties of modern sensors to selectively search neighbouring pixels. The key advantages of such an approach is through the constant time access to potential neighbours and we further show that the expected number of pixels examined for each kNN query is independent of the total image size. While the proposed approach can be adapted for various 3D sensors (Kinect-like, 3D LiDAR etc.) with simple projection models, we demonstrate the approach using stereo generated depth maps. Furthermore we show the k-nearest neighbour graph (kNNG) constructed using this approach is suitable for spatial clustering as shown in Fig. 1.

A key motivation for this work is in attempting to bring various sophisticated segmentation algorithms from the data mining community to robotics by addressing efficiency issues to meet real-time requirements. The work presented here was developed without knowledge of a similar approach described as *OrganizedNeighbor* search in the point cloud library (PCL) [Rusu and Cousins, 2011]. To the best of the author's knowledge a detailed analysis of the *OrganizedNeighbor* method isn't presented in any scientific publication. In this paper we use our independently developed method PANSearch which is based on the same projection principles as *OrganizedNeighbor* with the aim of highlighting scenarios where such an approach is advantageous over a common tree based efficient search.

The paper is organised as follows: The next section positions the proposed approach among existing works. Depth maps and kNNG are described in section 3, while section 4 details efficient kNNG construction. Section 5 describes how the kNNG is used for unsupervised segmentation. Section 6 shows the performance of the proposed method before a conclusion is given in section 7.

2 RELATED WORK

Unsupervised spatial clustering methods are used across many disciplines, including computer vision, pattern recognition, data mining and more recently robotics [Klasing *et al.*, 2008; Moosmann and Fraichard, 2010; Bewley *et al.*, 2011]. These methods are favourable as they do not require prior knowledge of the number of clusters in the scene and make no assumptions on the shape or convexity of the clusters. One of the most common spatial clustering algorithms is DBSCAN [Ester *et al.*, 1996] which selects unvisited points at random to begin growing a cluster by expanding along neighbouring points according to a minimum local density criteria. Defining a fixed minimum point density to construct clusters from 3D sensor data such as in [Klasing *et al.*, 2008] can lead to over segmentation for distant objects as point sparsity increases with increased distance from the sensor. Cluster methods based on kNN graphs [Jarvis and Patrick, 1973; Ertoz *et al.*, 2003; Pauling *et al.*, 2009] resolves this problem of finding clusters with variable density by considering the mutual neighbourhood between points. Typically, the most computationally expensive step in these clustering methods is constructing the kNNG of the data. When the input is large and dense, these approaches prohibit the real-time performance required in robotic sensing.

The neighbourhood search problem has been extensively studied with a large focus on algorithms that can accommodate higher dimensional data and numerous similarity metrics. Associating neighbouring pixels with neighbouring 3D points has been used in surface normal estimation [Strom *et al.*, 2010; Douillard *et al.*, 2011] based on the 4 and 8 connected neighbourhood structures borrowed from the computer vision community[Felzenszwalb and Huttenlocher, 2004] for segmenting image data. These methods do not account for scenarios where the nearest neighbours may not be directly adjacent to the query point as common in 3D data with noise. A more relevant subset of the literature is concerned with closest point searching commonly associated to point cloud registration problems [Jost and Hugli, 2003; Elseberg *et al.*, 2012]. The majority of literature regarding efficient nearest neighbour searching can be found in the pattern recognition and data mining domains. The algorithms closely related to this work can be categorised into: space partitioning [Warnekar and Krishna, 1979; Nievergelt *et al.*, 1984; Meagher, 1982], data partitioning [Friedman *et al.*, 1977; Kolahdouzan and Shahabi, 2004; Freund *et al.*, 2007], or dimensionality reduction [Friedman *et al.*, 1975; Andoni *et al.*, 2006; Connor and Kumar, 2009; Min *et al.*, 2010].

Space partitioning methods can be as simple as dividing the search space into a grid structure [Warnekar and Krishna, 1979; Nievergelt *et al.*, 1984] to enable efficient constant time access to points binned into a specific location bucket. These methods typically require an excessive amount of memory when high resolution grids are used to minimise the number of points in each bucket. Several memory efficient space partitioning methods based on octrees [Meagher, 1982] have been proposed which have variable resolution based on local point densities. These methods require prior knowledge of the optimal partitioning resolution for efficient kNN searching in non-uniform data density distributions.

The second category divides the search space along partitions defined by points from the dataset. Binary tree based methods such as the kd-tree are commonly

used to recursively split the data along axial aligned partitions with the median (for kd-trees [Friedman *et al.*, 1977] or bounding box (for R-trees [Guttman, 1984]) of each subset. This data structure allows for efficient spatial querying by cutting down on the number of candidate points when traversing the tree. Many variants of the kd-tree have been proposed to improve search times to find approximate nearest neighbours (ANN) using randomised trees [Arya *et al.*, 1998] and random projections [Freund *et al.*, 2007]. Other data defined partitioning methods construct a Voronoi representation of the entire dataset [Kolahdouzan and Shahabi, 2004] or recursive subsets in the form of a k-means tree [Nister and Stewenius, 2006; Wang, 2011]. Muja and Lowe have developed the fast library for approximate nearest neighbours (FLANN) [Muja and Lowe, 2009] which selects between a hierarchical k-means tree and randomised k-d tree algorithm with automatic parameter optimisation, dependent on the dataset. The authors reported an order of magnitude speed improvement over other state-of-the-art ANN search implementations and as a result it is now part of several robotics related software packages[1,2]. In our experiments we use this implementation as a basis of comparison in speed and precision.

The final category is dimensionality reduction based search methods. The various tree based partitioning methods listed above share the undesirable characteristic that the number of partitions checked for each NN query increases with the dataset size. Locality preserving functions can be used to evaluate a relative index into the data structure which is derived from the query position, providing constant access time to each partition. For example Connor and Kumar [Connor and Kumar, 2009] use the Morton Z-order which is synonymous to a depth first search of an octree.

Instead of using tree-like data structures an increasingly common form of dimensionality reduction is in the use of hashing functions that transform the $\mathbb{R}^d$ feature vectors into a single binary value used as an index. Singh and Singh [Singh and Singh, 2012] compute multiple two-dimensional projections of higher dimensional data with each projection relative to a different point before partitioning into a two dimensional grid of buckets. Local sensitivity hashing (LSH) methods [Lv *et al.*, 2007; Andoni *et al.*, 2006] map similar points to the same bucket with high probability. These methods typically require a substantial amount of memory and are better suited to range queries.

These methods all require a form of preprocessing of the data before performing nearest neighbour queries on the dataset. In this paper we eliminate this step by exploiting the implicit structure in depth maps produced by modern 3D sensors and propose an efficient projection based kNN search algorithm which does not scale in search complexity with the dataset size.

[1]http://www.ros.org/wiki/flann

[2]http://rock-robotics.org

3 Dense 3D Map and Graph Representations

Recent approaches to real-time 3D segmentation utilising 2D lasers [Klasing *et al.*, 2009] have inspired our work in extending this concept to dense 2.5D depth maps. Storing the depth data in a structured order corresponding to the scan angle, enables efficient spatial access to 3D data, constrained by the perspective geometry of the sensors. Fig. 2 shows the projection of a sphere (representing the upper limit of potential neighbours) onto the image plane using a projective camera model applied to the depth map.

A depth map is defined in this paper as a two dimensional grid of pixels such that each pixel represents the spatial distance along the z-axis to a 3D point. Using a projective camera model the x and y coordinates can be easily computed using the pixel's depth and grid location. By replacing the single depth value with the corresponding 3D coordinate at each pixel location while keeping the grid structure, we allow for efficient spatial indexing as opposed to storing the point cloud as a simple list and then computing a k-d tree.

For the remainder of this paper we use the notation of a single lower case character to represent a pixel index or offset in the discrete grid space such as $p \in \mathbb{Z}^2$ while upper case characters signify the corresponding 3D point $P \in \mathbb{R}^3$. The grid storing 3D points can be thought of as a $\mathbb{Z}^2 \mapsto \mathbb{R}^3$ mapping function such that $P = \mathcal{I}_{xyz}(p)$. Using this alternative representation has several desirable attributes for nearest neighbour searching, including: the data is already organised into a grid as provided by the sensor, the grid format enables constant time random access to each cell, each grid cell maps to only a single 3D point and the projective nature of the data preserves 3D neighbourhoods in the 2D grid.

The kNNG construction problem can be defined as follows: given a set of n points where $P_i \in \mathbb{R}^3 \{i = 1 \dots n\}$, construct a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each vertex corresponds to a pixel in the depth map and $\mathcal{E}$ is the set all pairs $\mathcal{E} : (P_i, P_j), i \neq j$ such that P_j is one of the k-nearest neighbours of P_i. A naive approach to finding the k-nearest neighbours to a given point P_i, is to search over every other point P_j keeping only the k points with the shortest distance. Constructing a kNNG using this approach has a time complexity of $\mathcal{O}(n^2)$. Now that these concepts have been introduced, we proceed to describe the PAN-Search algorithm.

4 PAN-Search: Projected Area Neighbourhood Searching

When dealing with dense depth maps as an input for kNNG based clustering, the dense 3D map representation enables efficient neighbourhood searching regardless of the input image size. In this section we present a simple search algorithm for finding kNN in 3D metric space for any given query point Q positioned at q in the dense 3D map $\mathcal{I}_{xyz}$.

Algorithm 1 shows the key steps for finding the kNN from a given location q in the dense 3D map $\mathcal{I}_{xyz}$. The critical concept behind this algorithm is exploiting the characteristic of projection models that the kNN points to Q are projected near to q in grid space. By initially selecting k candidate neighbours the 3D search space is bound to the kth closest point P_k shown in Fig. 2. The grid cells covered by the projection of this boundary are searched in an order designed to rapidly meet the terminating criteria described below.

Algorithm 1 Projected Area Neighbourhood Searching

Input: $\mathcal{I}_{xyz}$ ▷ each pixel is a 3D point
Input: q
Input: k
Input: f ▷ focal length in pixels
Input: Search Strategy *search_list*
Output: kNN

function PAN-SEARCH($\mathcal{I}_{xyz}, q, k, f, search_list$)
  $Q \leftarrow \mathcal{I}_{xyz}(q)$
  for $i = 1$ to $\|search_list\|$ **do**
    $r_o \leftarrow \|o_i\|$ ▷ where o_i is the pixel offset
    $P \leftarrow \mathcal{I}_{xyz}(q + o_i)$
    $d \leftarrow \|P - Q\|$
    $kNN.insert(o_{ij}, d, k)$
    if $kNN.size() > k$ **then**
      $kNN.sort()$ ▷ sort by d
      $kNN.pop()$ ▷ remove largest d
      $r_{max} \leftarrow updateBoundary(kNN_k, \mathcal{I}_z(q), f)$
      if $(r_o > r_{max})$ **then return** kNN
      end if
    end if
  end for
end function

4.1 Image Search Pattern

The key to rapidly narrowing the search space is in the order in which neighbouring pixels are visited to seek the k closest points. The projection characteristic that neighbouring points on an object's surface are projected to a neighbouring location in the grid space we can efficiently traverse radially outward from the given query point q. The efficiency of this approach is degraded as the query point approaches the boundary of an object. As the ratio of boundary points to non-boundary points is small with real world depth maps the overall effect of boundary points is minimal.

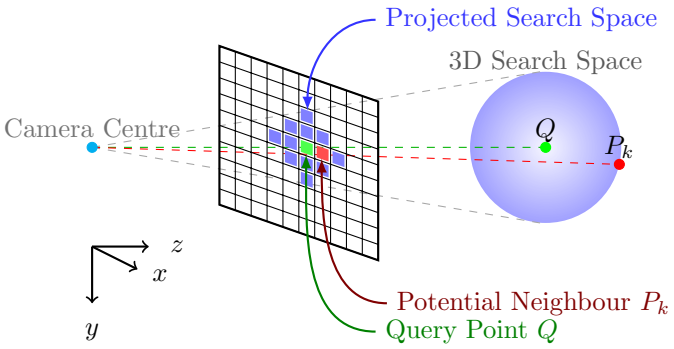
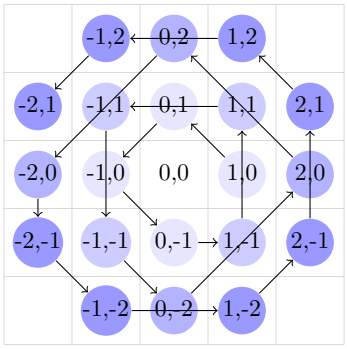


Figure 2: Illustration of the projected image search space from a sphere centred at the query point Q and radius of $\|QP_k\|$ where P_k is the kth closest potential neighbouring point found so far for Q.



1,0 → 0,1 → -1,0 → 0,-1 → 1,-1 → 1,1 → -1,1 → -1,-1 → 0,-2 → 2,0 →

Figure 3: The index offsets for the first 20 candidate neighbours the in the fixed search pattern. The colour intensity of each node indicates the radial distance to the query position q.

The order in which neighbouring pixels are prioritised is determined by their proximity to the query pixel q. Fig 3 shows a potential sequence of index offsets to be applied to q in a radially growing search pattern. A simple yet efficient search strategy is to sort the pixel index offsets by their radius to the query pixel with an arbitrary order selected for offsets with equivalent radius in the image. This essentially forces points projected closer in the grid space to be searched completely before expanding the search radius. The sorting of these offsets can be computed offline as this pattern is kept constant and data independent for efficient online performance.

4.2 Terminating Criteria

Once a minimum of k valid pixels are considered, any point closer than P_k would also need to lie in the projected search boundary. As closer points are discovered using the radially expanding search pattern, the distance from the query point Q to the kth closest point decreases, contracting the projected search area towards Q. The search for kNN terminates when all pixels in the projected boundary with radius defined by the kth closest point has been checked.

While the true perspective projection of a sphere into the image plane casts an ellipse, we argue that a simple approximation of a circle around q as the image boundary saves on complex computation while maintaining reasonable precision in finding the true nearest neighbours. The maximum radius of the projected search boundary $\mathcal{B}_q$ in grid space for a camera like sensor is given by:

$$r_{max} = \frac{\|QP_k\|.f}{|Q_z|} \tag{1}$$

where $\|QP_k\|$ is the radius of the bounding sphere, f is the focal length in pixels and Q_z is the depth at q. The terminating condition to finish the search is when all pixels $p \in \mathcal{B}_q$ have been considered such that $\|pq\| \leq r_{max}$ (see line 11 and 12 of Algorithm 1).

Additionally a bailout condition can be set by limiting the size of the search pattern to terminate the search after a maximum number of pixel offsets are considered. This bailout can be used to simultaneously identify outliers while capping the maximum time spent searching for neighbours of outlier points.

Fig. 4 shows the relative number of pixels considered at each point in the depth map (shown in Fig. 1) with white pixels near object boundaries representing points where the bailout condition was met before the projected area could contract to the furthest offset in the search pattern. As expected, pixels around object boundaries take longer to compute compared to smooth continuous surfaces as the initial candidate neighbourhood set starts with points from both the foreground and background. Another interesting observation is that surfaces with high gradients relative to the view point carry high search cost while surfaces more parallel to the image plane have low search cost.

4.3 Search Complexity

The performance of this algorithm in terms of time is dependent on the data stored in the dense 3D map, with a worst-case complexity of $\mathcal{O}(l)$, where l is the maximum length of our ordered search pattern. However with the exception of occluded boundaries and discontinuities it is expected that the physical proximity of a given point is also proximal in the image grid space. Using a radially expanding search pattern, the expected run time is $\mathcal{O}(k)$ for points on smooth surfaces as opposed to k-d tree searching which is $\mathcal{O}(klog(n))$ where $1 \leq k \ll n$.

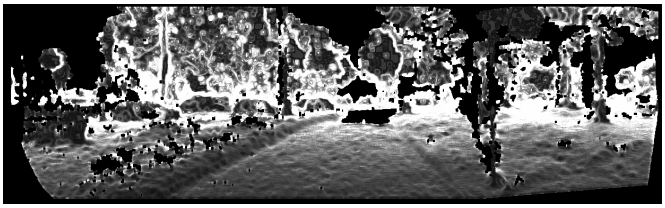


Figure 4: Heat map of where pixels with high run-times are located. The white (hottest) pixels coincide with the bailout condition.

5 Spatial Clustering using a kNN Graph

The kNNG can now be used as a basis for unsupervised segmentation in a region growing framework that spatially clusters the dense 3D input data. It is important to note that the kNNG in this raw form is directional as the kNN relationship is asymmetric, making the resulting clusters dependent on the initial seeding. To overcome this, we only expand a cluster along edges where both vertices are mutual neighbours, effectively cutting asymmetric edges. The resulting undirected graph enables deterministic clustering results with randomly selected seeds. For more details on this clustering method, the reader is encouraged to refer to [Jarvis and Patrick, 1973; Ertoz *et al.*, 2003].

This form of clustering is aided with knowledge of object boundaries discovered using the early bailout condition. As the resulting kNN points found up to the bailout are not guaranteed, we consider the corresponding vertex to be invalid and remove it from the graph. This not only prevents clusters growing across invalid neighbourhood edges, but also aids the clustering, as these points typically lie on an object's boundary.

6 Experiments and Results

The proposed approach is evaluated by comparing the speed and correctness against an efficient k-d tree implementation. The k-d tree software used in this evaluation is the OpenCV (version 2.4.3) implementation of FLANN (version 1.6.11), which can also be found PCL [Rusu and Cousins, 2011]. The k-d tree construction and search parameters for FLANN are left as default in all tests unless stated otherwise.

6.1 Datasets

We evaluate the performance of our system using 20 depth images derived from the ground truth disparity maps of the 2006 Middlebury dataset [Scharstein and

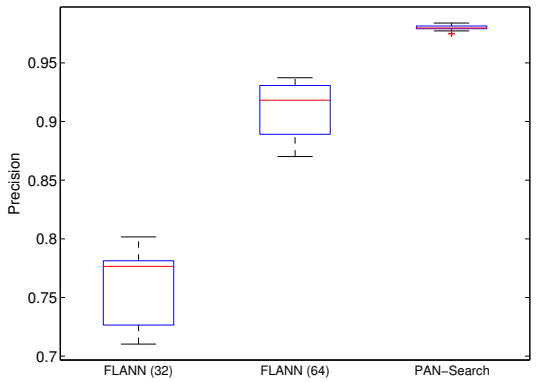


Figure 5: Average precision of 10 NN over 20 depth images using ELAS on the KITTI Dataset.

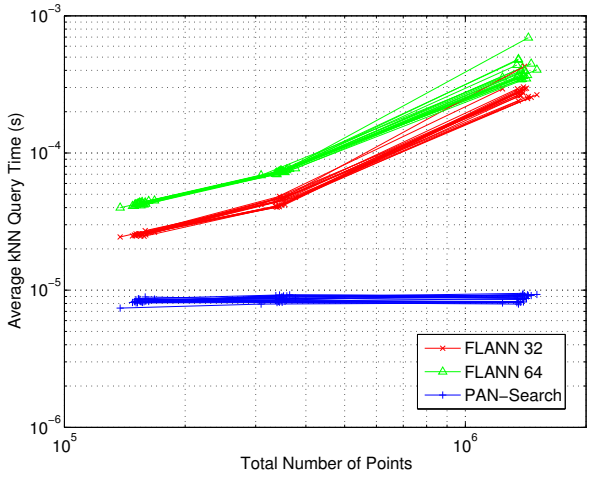


Figure 6: Average kNN query time (k=10) for 20 images from the Middlebury 2006 dataset using three different resolutions. Best viewed in colour.

Pal, 2007]. This dataset was selected as it contains various dense disparities at three difference resolution settings. While this provides a convenient benchmark to measure the speed performance for different input sizes (shown in Fig. 6) it lacks spatial variety as all stereo images are of indoor scenes or close objects. We address this by additionally comparing the performance of the proposed method using depth images generated from a stereo vision system of an outdoor traffic scene. For this we selected 20 frames of the KITTI dataset [Geiger *et al.*, 2012] from various sequences which contain several objects of potential interest such as cars and pedestrians. The stereo image pairs were converted to dense depth images using the Efficient LArge-scale Stereo (ELAS) matching algorithm published in [Geiger *et al.*, 2011] along with the stereo parameters from [Geiger *et al.*, 2012]. An example of the depth image generated using ELAS is shown in Fig. 1.

6.2 Correctness Evaluation

The precision of the proposed method is compared to that of FLANN with default search parameters of 32 maximum checks per query and 64 maximum checks. The results of applying these algorithms to the 20 depth images from the KITTI dataset are shown in Fig. 5. The exact nearest neighbours for each valid pixel were found using a brute force linear search considering potential neighbours across the entire image. Precision here is calculated as the number of points $P \in \mathcal{Q}_{knn}$ with $\|PQ\| \leq \|P'_kQ\|$ where P'_k is the true kth nearest neighbour from the brute force search. This experiment supports the projected circle approximation of the spherical boundary to limit the search space in PAN-Search.

6.3 Timing

A prototype of the PAN-Search algorithm was implemented in C++ and was deployed on a single core of a 2.2GHz Intel i7 processor with 8GB of RAM. OpenCV 2.4[3] was used for storing the neighbourhood graph as a matrix while the *priority_queue* from the standard template library is used for maintaining an ordered list of nearest neighbours for kNN searching. The performance of the proposed kNN graph construction algorithm was compared to the OpenCV implementation of FLANN on the same test computer.

Fig. 6 shows that PAN-Search's kNN time is independent of the total size of the input cloud as its search space is defined by the local neighbourhood, which is accessible in constant time due to the grid structure of the dense 3D image. The total time (in seconds) to construct the kNN graph for 20 frames from the KITTI datasets is shown in Fig. 7 where the value of k ranges from 1 to 20. Our $\mathcal{O}(k)$ method outperforms FLANN with 32 and 64 checks on this dataset up to $k = 14$ and 20 respectively. It is important to note that the recorded times is only for kNN graph construction and doesn't include the preprocessing time to construct the internal k-d trees used by FLANN. The k-d tree preprocessing took an average of 0.5 and 2.2 seconds on the KITTI and full resolution Middlebury datasets respectively.

7 Conclusion

In this paper we have shown a way of exploiting the 2.5D nature of dense depth maps by utilising the projection properties of modern sensor data to selectively

[3] http://opencv.org/

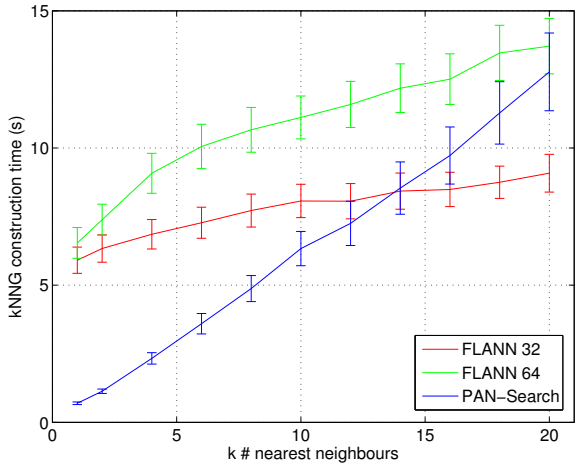


Figure 7: Run time vs. k showing mean and standard deviation for 20 frames from the KITTI traffic dataset. Note the y-axis represent the total time to construct the kNNG over an 0.5 Mega-pixel depth image.

search neighbouring pixels until no potential candidates remain. This provides an alternative solution to efficiently finding the k nearest neighbours to a set of given query points from a dense depth map with known camera projection matrix. This approach demonstrates a significant speed-up over common tree based search methods as it maintains a constant search time per pixel regardless of image size. Furthermore we experimentally evaluate range of k values where this method outperforms the ubiquitous k-d tree method. Additionally each kNN query can be computed in parallel providing a further speed-up on what is presented here. We believe that this is an important step towards constructing kNN graphs in real-time and ultimately real-time unsupervised segmentation of dense 3D data. In future work we intend on using other sources of information in addition to the 3D spatial location for constructing clusters based on the kNN graph connectivity.

Acknowledgments

This research has been supported by the Australian Coal Association Research Program (ACARP). The authors also would like to thank Kylie He and Peter Corke for valuable discussions and useful feedback regarding this work.

References

[Andoni *et al.*, 2006] A Andoni, M Datar, N Immorlica, P Indyk, and V Mirrokni. Locality-sensitive hashing using stable distributions. In *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*, chapter 4, pages 55–67. MIT Press, 2006.

[Arya *et al.*, 1998] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, November 1998.

[Bewley *et al.*, 2011] Alex Bewley, Rajiv Shekhar, Sam Leonard, Ben Upcroft, and Paul Lever. Real-time volume estimation of a dragline payload. In *2011 IEEE International Conference on Robotics and Automation*, pages 1571–1576, Shanghai, China, May 2011. IEEE.

[Connor and Kumar, 2009] Michael Connor and Piyush Kumar. Fast construction of k-nearest neighbor graphs for point clouds. *IEEE transactions on visualization and computer graphics*, 16(4):599–608, 2009.

[Douillard *et al.*, 2011] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel. On the segmentation of 3D LIDAR point clouds. In *2011 IEEE International Conference on Robotics and Automation*, pages 2798–2805. IEEE, May 2011.

[Elseberg *et al.*, 2012] Jan Elseberg, Stéphane Magnenat, Roland Siegwart, and Nüchter Andreas. Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration. *Journal of Software Engineering for Robotics (JOSER)*, 3(1):2–12, 2012.

[Ertoz *et al.*, 2003] Levent Ertoz, Michael Steinbach, and Vipin Kumar. Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data. In *Second SIAM International Conference on Data Mining*, 2003.

[Ester *et al.*, 1996] Martin Ester, Hans-peter Kriegel, Jörg Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *2nd International Conference on Knowledge Discovery and Data Mining*, 1996.

[Felzenszwalb and Huttenlocher, 2004] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181, September 2004.

[Freund *et al.*, 2007] Yoav Freund, Sanjoy Dasgupta, Mayank Kabra, and Nakul Verma. Learning the structure of manifolds using random projections. *Neural Information Processing Systems*, 20:537, 2007.

[Friedman *et al.*, 1975] J.H. Friedman, F. Baskett, and L.J. Shustek. An Algorithm for Finding Nearest Neighbors. *Computers, IEEE Transactions on*, C-24(10):1000–1006, 1975.

[Friedman *et al.*, 1977] Jerome H Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Trans. Math. Softw.*, 3(3):209–226, 1977.

[Geiger *et al.*, 2011] Andreas Geiger, Martin Roser, and Raquel Urtasun. Efficient Large-Scale Stereo Matching. In *Proceedings of the 10th Asian conference on*

Computer vision - Volume Part I, pages 25–38, Queenstown, New Zealand, 2011. Springer-Verlag.

[Geiger *et al.*, 2012] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. Ieee, June 2012.

[Guttman, 1984] Antonin Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data - SIGMOD '84*, page 47, New York, New York, USA, 1984. ACM Press.

[Jarvis and Patrick, 1973] R.a. Jarvis and E.a. Patrick. Clustering Using a Similarity Measure Based on Shared Near Neighbors. *IEEE Transactions on Computers*, C-22(11):1025–1034, November 1973.

[Jost and Hugli, 2003] Timothée Jost and H. Hugli. A multi-resolution ICP with heuristic closest point search for fast and robust 3D registration of range images. In *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pages 427–433. IEEE, 2003.

[Klasing *et al.*, 2008] Klaas Klasing, Dirk Wollherr, and Martin Buss. A clustering method for efficient segmentation of 3D laser data. In *IEEE International Conference on Robotics and Automation*, pages 4043–4048. Ieee, May 2008.

[Klasing *et al.*, 2009] K. Klasing, D. Wollherr, and M. Buss. Realtime segmentation of range data using continuous nearest neighbors. In *IEEE International Conference on Robotics and Automation*, pages 2431–2436. Ieee, May 2009.

[Kolahdouzan and Shahabi, 2004] Mohammad Kolahdouzan and Cyrus Shahabi. Voronoi-based K nearest neighbor search for spatial network databases. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, VLDB '04, pages 840–851. VLDB Endowment, 2004.

[Lv *et al.*, 2007] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-probe LSH: efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd international conference on Very large data bases*, pages 950–961, 2007.

[Meagher, 1982] D Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147, 1982.

[Min *et al.*, 2010] Kerui Min, Linjun Yang, John Wright, Lei Wu, Xian-Sheng Hua, and Yi Ma. Compact projection: Simple and efficient near neighbor search with practical memory requirements. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3477–3484. Ieee, June 2010.

[Moosmann and Fraichard, 2010] Frank Moosmann and Thierry Fraichard. Motion estimation from range images in dynamic outdoor scenes. In *2010 IEEE International Conference on Robotics and Automation*, pages 142–147. IEEE, May 2010.

[Muja and Lowe, 2009] Marius Muja and David G Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[Nievergelt *et al.*, 1984] J. Nievergelt, Hans Hinterberger, and Kenneth C. Sevcik. The Grid File: An Adaptable, Symmetric Multikey File Structure. *ACM Transactions on Database Systems*, 9(1):38–71, January 1984.

[Nister and Stewenius, 2006] D Nister and H Stewenius. Scalable Recognition with a Vocabulary Tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)*, volume 2, pages 2161–2168. IEEE, 2006.

[Pauling *et al.*, 2009] Frederick Pauling, Michael Bosse, and Robert Zlot. Automatic Segmentation of 3D Laser Point Clouds by Ellipsoidal Region Growing. In *Australasian Conference on Robotics and Automation (ACRA)*, 2009.

[Rusu and Cousins, 2011] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4. IEEE, May 2011.

[Scharstein and Pal, 2007] Daniel Scharstein and Chris Pal. Learning Conditional Random Fields for Stereo. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, June 2007.

[Singh and Singh, 2012] Vishwakarma Singh and Ambuj K. Singh. SIMP. In *Proceedings of the 15th International Conference on Extending Database Technology - EDBT '12*, page 492, New York, New York, USA, 2012. ACM Press.

[Strom *et al.*, 2010] J Strom, A Richardson, and E Olson. Graph-based segmentation for colored 3D laser point clouds. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2131–2136. IEEE, October 2010.

[Wang, 2011] Xueyi Wang. A fast exact k-nearest neighbors algorithm for high dimensional search using k-means clustering and triangle inequality. In *The 2011 International Joint Conference on Neural Networks*, volume 43, pages 1293–1299. IEEE, July 2011.

[Warnekar and Krishna, 1979] C.S. Warnekar and G. Krishna. A heuristic clustering algorithm using union of overlapping pattern-cells. *Pattern Recognition*, 11(2):85–93, January 1979.