

# Structure Tensors for General Purpose LIDAR Feature Extraction

Yangming Li<sup>1,2</sup> and Edwin B. Olson<sup>2</sup>

1, Institute of Intelligence Machines, Chinese Academy of Sciences, Hefei, Anhui, 230031

2, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109

Email: ymli@umich.edu, ebolson@umich.edu

<http://april.eecs.umich.edu>

**Abstract**—The detection of features from Light Detection and Ranging (LIDAR) data is a fundamental component of feature-based mapping and SLAM systems. Classical approaches are often tied to specific environments, computationally expensive, or do not extract precise features.

We describe a general purpose feature detector that is not only efficient, but also applicable to virtually any environment. Our method shares its mathematical foundation with feature detectors from the computer vision community, where structure tensor based methods have been successful. Our resulting method is capable of identifying stable and repeatable features at a variety of spatial scales, and produces uncertainty estimates for use in a state estimation algorithm. We verify the proposed method on standard datasets, including the Victoria Park dataset and the Intel Research Center dataset.

**Index Terms**—Robot navigation, SLAM, LIDARs, Feature detection, Corner Detector

## I. INTRODUCTION

The Simultaneous Localization and Mapping (SLAM) problem is generally regarded as the key problem in mobile robotics, because locations and environmental information are the two important factors to mobile robots. Laser Range Finders (also known as LIDARs) are widely used in many applications, because of their ability to accurately measure both bearing and range to objects and their robustness to environmental inference. In the SLAM field, there are mainly two types of mapping approaches using LIDARs: pose-based algorithms and feature-based algorithms. Conceptually, the pose graph resulting from pose-to-pose matches (produced by a LIDAR scan matcher, for example) resemble a feature-based graph in which all the features have been marginalized out. Unfortunately, this marginalization creates dense graphs which slow modern SLAM algorithms.

For example, suppose that a particular object is visible from a large number of poses. In a scan matching approach (in which landmarks are not part of the state vector), this will lead to constraints between each pair of poses: the graph becomes fully connected and has  $O(N^2)$  edges. In contrast, a feature based approach has only  $O(N)$  edges.

In the case of sparse Cholesky factorization, Dellaert showed that the optimal variable reordering is *not* necessarily the one in which features are marginalized out first [1]: the information matrix can often be factored faster when there are landmarks. Similarly, the family of stochastic gradient descent (SGD) algorithms [2], [3] and Gauss-Seidel relaxation [4], [5] have runtimes that are directly related to

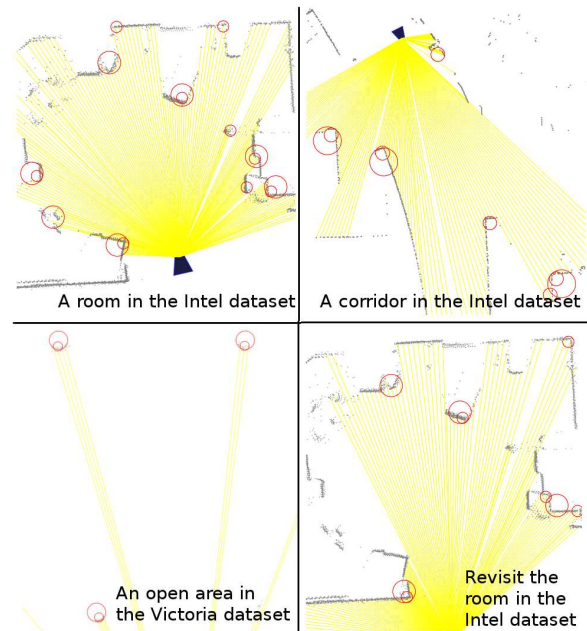


Fig. 1. Multi-scale normal structure tensors feature extraction. Red circles indicate extracted features; size of circles indicates the scale of the feature. Blue triangles denote robots; yellow lines denote current observations and yellow points are accumulated observation points.

the number of edges. The extra edges also frustrate sparse information filters, such as SEIFs [6] and ESEIFs [7], [8].

Additionally, data association in feature-based methods is computationally efficient, because searching over landmarks is less expensive than searching over the space of rigid-body transformations. The cost of scan matching, for example, can become prohibitively expensive when the prior uncertainty is very large. While scan matching algorithms with large search windows can be implemented efficiently [9], the computational complexity of feature-based matching is nearly independent of initialization error. Finally, feature-based methods tend to work better outdoors because they often reject ground strikes that result when the robot pitches or rolls.

In short, feature-based methods would be preferable to scan matching if they were able to offer the same flexibility as scan-matching approaches.

Classical feature detectors exploit the characteristics of an environment: in indoor settings, lines, corners and curves have been used [10], [11], [12], [13], [14]. Outdoors, the hand-tuned tree detector originally developed for the Victoria Park dataset [15], has been used almost universally (see [16], [8], [17] for representative examples). Naturally, tree detec-

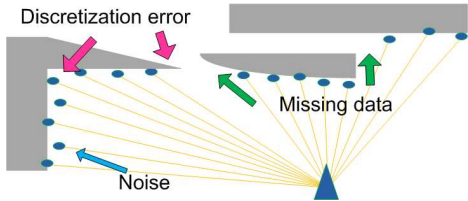


Fig. 2. Challenges in feature extraction from LIDAR data. Gray shapes denote obstacles and blue points denote observations. Three problems are indicated with arrows.

tors work poorly in offices, and corner detectors work poorly in forests. The lack of a general-purpose feature detector that works well in varied environments has been an impediment to robust feature-based systems.

Recent work has addressed this problem; Pedraza, *et al.* [18] use B-Splines to represent unstructured environments. Their general-purpose method helps to improve the applicability of feature-based methods to a wider variety of environments. However, the segmentation of laser data, the selection of control points, and the feature representation in the data association process are still areas of active research.

Zlot and Bosse [19] propose a number of heuristics for identifying stable keypoints in LIDAR data. Their methods begin with clustering connected components and then either 1) computing the centroids of each segment, 2) computing the curvature of each segment, or 3) iteratively computing a locally-weighted mean position until it converges. They also examined descriptors which aid data association. Li and Olson proposed a detector that can be seen as replacing the three mechanisms with a single method [20]. They convert both 2D and 3D laser data into images and extract features using the multi-scale Kanade-Tomasi corner detector. The method is virtually applicable to any environment, but its computation complexity is proportional to the size of the converted image, and incorporates some processing steps that, while effective, are difficult to analyze rigorously.

In this paper, we describe a general-purpose feature detector that extracts repeatable and stable features in virtually any environment. The central contributions of this paper are:

- We propose a general-purpose feature detector for LIDAR data by computing the structure tensors of surface normals.
- We show how to quickly extract stable and repeatable features in the presence of noise, discretization error and missing data.
- We demonstrate that our method has better performance in varied environments than earlier approaches.

We begin by describing the challenges in feature extraction from LIDAR data, and then describe the proposed method in detail. In Section III, we address the computational complexity of the proposed method and evaluation metrics for feature detectors. In Section IV, we present experimental evaluations of our method versus earlier approaches.

## II. STRUCTURE TENSOR OF NORMALS FEATURE DETECTOR

### A. Method Overview

Feature extraction from LIDAR data is difficult due to the following challenges (see Fig. 2):

- **Sensor noise.** LIDAR data is inevitably contaminated by noise. This noise, if not managed, can create false positives.
- **Discretization error.** Sampling the world at a fixed angular resolution causes distant parts of the world to be sampled very coarsely. This range-dependent resolution can make it difficult to recognize the same environment from different ranges.
- **Missing data.** Occlusions and reflective objects result in gaps in sensor data. A feature detector must be robust to these gaps.

To address these challenges, our method combines three steps:

- **Pretreatment.** We group raw observations into contours and smooth points in each contour with a probabilistically rigorous method. We then compute the surface normals along the contours.
- **Candidate detection.** We slide a circular window around the contours, computing the structure tensor of the surface normals within the window. The minimum eigenvalue of the structure tensor measures the feature strength; strong features become feature candidates.
- **Candidate suppression.** We reject feature candidates in the vicinity of stronger candidates in order to reduce feature-matching confusion.

The three steps in the proposed method are explained in detail in following sub-sections.

### B. Pretreatment

Like most LIDAR feature extractors, we begin by grouping together points that likely belong to the same physical surface. These “contours” are grown agglomeratively by connecting nearby points that are within some threshold distance. We process the area around each contour separately, which eliminates wasted computation that would result from searching empty areas. Processing them separately also eliminates false positives due to physically nearby but separate surfaces.

The noise in LIDAR data is a major concern; left unchecked, range noise would generate a large number of false positives. We propose a principled probabilistic filter to reduce noise. This step incorporates a prior on the smoothness of the surface and computes the maximum likelihood position of the noise given the prior and the noisy samples. The cost associated with individual LIDAR points is derived directly from the sensor’s noise model: moving the  $i$ -th observation point away from its observed point by a vector  $\delta_i$  incurs a cost  $\delta_i^T \Sigma_i^{-1} \delta_i$ , where  $\Sigma_i$  is the covariance associated with the observation.

The cost associated with surface smoothness is formulated as a quadratic penalty on the angle between each line segment. Let  $\beta$  be a parameter that reflects a prior on the smoothness of surfaces and  $\theta_i$  be the orientation of the normal between point  $i-1$  and  $i$ . The total cost can then be written:

$$\chi^2 = \delta_i^T \Sigma_i^{-1} \delta_i + \beta (\theta_i - \theta_{i-1})^2 \quad (1)$$

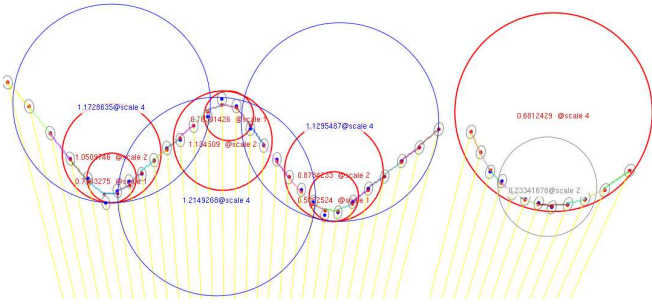


Fig. 3. Overview of the proposed method. Blue points are raw observations and red points denote the smoothed counterparts; yellow lines denote observation angles and colored short lines are composed of grids; small gray ellipses denote observation uncertainties; red circles indicate extracted features, while blue and gray circles denote feature candidates suppressed for by strength comparison and distance comparison, respectively. Numbers denote feature strengths and corresponding scales.

We solve this optimization problem using relaxation [4]. With this approach, we compute improved positions for each point while fixing the positions of the points around it. This process is repeated until the process converges, which typically occurs within a few iterations. We have empirically noted that the smoothing filter produces similar (and good) results over a range of values of  $\beta$  (from 0.5 to 5).

The LIDAR component of the cost function can be simplified if we ignore the angular noise. We begin with the usual LIDAR noise model [21]:  $Q \sim \mathcal{N}(\mu_Q, \Sigma_Q)$

$$\mu_Q = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \Sigma_Q = \begin{bmatrix} \sigma_r^2 & \sigma_{r\alpha} \\ \sigma_{r\alpha} & \sigma_\alpha^2 \end{bmatrix}$$

$\sigma_r$  and  $\sigma_\alpha$  indicate the range and angle standard deviations, respectively;  $\sigma_{r\alpha}$  denotes the covariance between range and angle. In comparison to the range noise, the angular noise is typically negligible [13]. This allows us to approximate Eqn (1) as  $\Delta r^2 / \sigma_r^2 + \beta(\theta_i - \theta_{i-1})^2$ , where  $\Delta r$  indicates the innovation on range. This simplification modestly accelerates the relaxation optimization.

Once points have been filtered, we compute the orientation of the surface normal along the lines connecting each point. We rasterize these normal directions into a grid map, where each cell in the gridmap stores an orientation. Bresenham’s line algorithm [22] is used to efficiently identify the grid cells that fall on the line between each pair of LIDAR points.

### C. Candidate Detection

At this point in the method, we’ve generated a rasterized image (or grid map). The cells in this grid map encode the surface normal direction at each point along the LIDAR contours. Grid cells that are not near a physical surface have the zero vector written to them instead.

Our goal is now to identify parts of this grid map that are repeatably, reliably, and accurately localizable. The key idea of our algorithm is to compute the structure tensor corresponding to local neighborhoods of this gridmap. The structure tensor measures the “diversity” of the normal directions in an area; areas with a variety of surface normals have stronger structure tensors than areas where the normals are more uniform (or where there are no surfaces at all).

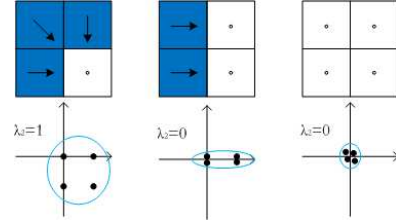


Fig. 4. Explanation of the structure tensor based feature detection. Top: Three representative shapes of corners with arrow indicating gradients. Bottom: The corresponding responses in coordinates and the minimum eigenvalues,  $\lambda_2$ .

This method of building the structure tensors is similar to that encountered in image processing, and in our earlier LIDAR feature detection paper. In both of those cases, the structure tensor is constructed from gradients computed from the grid map image itself (which contains intensities). In contrast, the grid map in this paper contains normals (analogous to the gradients in the previous work) which have been directly computed from the underlying laser data. As we will demonstrate, this new approach extracts significantly more stable features, in large part due to the fact that the underlying normals are more accurate [23].

To compute the structure tensor in a neighborhood of the gridmap, we enumerate the surface normals within the region and compute the outer product. We use a circular neighborhood in order to maximize the isotropic behavior of the detector [24]. Let  $n_i$  be a normal vector within the circular neighborhood. The structure tensor is then:

$$A = \sum_i \begin{bmatrix} n_{i,x}^2 & n_{i,x}n_{i,y} \\ n_{i,x}n_{i,y} & n_{i,y}^2 \end{bmatrix} \quad (2)$$

The structure tensor can be understood intuitively as the covariance matrix of the gradient directions (see Fig. 4). Suppose each the endpoint of each normal (starting from the origin) is plotted, and the sample covariance computed. When the normals point in different directions, the covariance matrix will be “fatter”; when the normals are distributed in a more uniform way (or when there are no normals nearby), the resulting covariance matrix is small.

The precise meaning of what it means for  $A$  to be “big” or “small” is the subject of some debate. Harris advocated a computationally inexpensive measure [25]:

$$C = |A| - \kappa \text{trace}(A)^2 \quad (3)$$

Shi and Tomasi argue that it is better to use the smallest eigenvalue of  $A$  as the corner strength function as [23]:

$$C = \min(\lambda_1, \lambda_2) \quad (4)$$

In the proposed method, we adopt the minimum eigenvalue as the indicator; our empirical experience indicates that is more precise measure than Eqn. (3) [23].

Since useful navigational landmarks can be both large (an irregularly shaped bush, perhaps) or small (a corner in a building), it is important that a feature detector be able to detect features over a range of scales. In the computer vision community, the standard procedure is to compute a image pyramid, in which the resolution of the image is geometrically reduced and the detection process repeated at

every level. A disadvantage of this approach is that positional precision of features detected at large scales is quite low because the spatial resolution of the image has been reduced.

We take a different approach. We extract features at different scales, but we do this by increasing the size of the structure tensor window, rather than reducing the resolution of the underlying grid map. The tensor window, regardless of its size, is evaluated at every pixel, giving us full-resolution position information for landmarks (even those extracted at large scales).

Normalization of the structure tensor becomes an issue when working across multiple scales. As the size of the structure tensor window increases, it naturally tends to contain more surface normals. Without normalization, for example, the best feature in the entire scene would be one that includes the entire scene. To encourage features that are spatially compact, we normalize the structure tensors by the diameter of the window. We experimented with other normalization factors (such as the *area* of the window, and the *number of normals* within the window), but these were out-performed by the diameter.

An interesting difference between computer vision and LIDAR methods is that vision methods must be scale invariant; LIDAR, on the other hand, directly acquires scale information. The scale information computed by our algorithm can be used as a descriptor to help reduce false matches. In other words, unlike computer vision, matches between features should only be allowed when they are at the same scale.

#### D. Candidate Suppression

As in computer vision feature detectors, an important step is to suppress features of low saliency. A cluster of features in close proximity are potentially confusing and ambiguous, for example.

First, local maximum suppression is performed for each scale: only points whose response is stronger than their neighbors are selected. Second, features are suppressed across scales: only features that are  $K$  times as strong as other features located within the same window are retained. The factor  $K$ , typically 2 in our experiments, inhibits features that are similar to other very nearby features. In turn, this improves data association performance.

Fig. 3 visually explains the proposed method. In the figure, blue points are raw observations in one contour and red points denote the smoothed counterparts; yellow lines denote observation angles and colored short lines are composed of grids; red circles indicate extracted features, while blue and gray circles denote feature candidates suppressed for strength and distance, respectively. Both features and candidates are labeled with corresponding strengths and scales. Fig. 1 shows extractions in both indoor and outdoor environments.

### III. DISCUSSION

#### A. Uncertainty Estimation

Feature uncertainties are of critical importance to SLAM applications. It has been previously shown that the covariance

of a structure tensor is proportional to the inverse of the structure tensor matrix [26].

We evaluated the accuracy of the covariance estimates of our features using real data from the Intel Research Center and Victoria Park datasets. Using a ground-truthed dataset, we were able to measure the error associated with every feature detection. Collecting this data over thousands of feature detections, we constructed a histogram of Mahalanobis distances. As shown in Fig. 8, the distribution of Mahalanobis distances fits the expected Chi-squared distribution.

#### B. Computation Complexity

The proposed method has low computation complexity, largely in part to our ability to prune the search to only a small portion of the space around the robot. In particular, significant speedups can be obtained by searching for features only near contours.

We can also examine the asymptotic complexity of the processing steps that comprise our method. The computation complexity of the contour extraction is  $O(N \log N)$  [13]. The contour smoothing operation runs in  $O(N)$  per iteration; typically, only a few iterations are required.

Computing normals and rasterizing the grid map requires  $O(N + M/\epsilon)$ , where  $N$  is the number of points,  $M$  is the perimeter formed by those points, and  $\epsilon$  is the grid size. The structure tensors require a potentially large amount of time; for every pixel in the grid map, all pixels within a radius must be tallied. The precise amount of time depends on the range of scales, grid map resolution, etc. The computational cost of candidate suppression similarly depends on the local density of features; all pairs of nearby candidates must be considered.

#### C. Repeatability Evaluation

In a SLAM context, repeatability (the consistency with which a given landmark is observed) is critical. Each re-observation of a landmark creates a “loop closure”, improving the quality of the posterior map.

Although there is a vast body of work on feature detection, there is much less on the detector evaluation. Mohannah and Mokhtarian [27] define the consistency of corner numbers as  $CCN = 100 \times 1.1^{|n_w - n_o|}$ , where  $n_o$  is the number of features in an image and  $n_w$  is the number in the warped counterpart. Trajkovic and Hedley [28] define stability as the ratio of strong matches over the total number of features. Schmid, *et al.* [29] define the repeatability as the number of points repeated between two images with respect to the total number of detected points, as:  $r_i(\epsilon) = \frac{R_i(\epsilon)}{\min(n_1, n_i)}$ ,  $R_i(\epsilon)$  denotes the number of repeated features, and  $n_1, n_i$  denote the number of features detected in the common part of two images. Rosten and Drummond [24] propose a similar method as Schmid except they adopt a 3D surface model to compute where detected features should appear.

These evaluation methods from the computer vision field can not be directly applied on our method, because they either need human labelled features [27] and a tracking algorithm [28], or require adjusting the number of features to

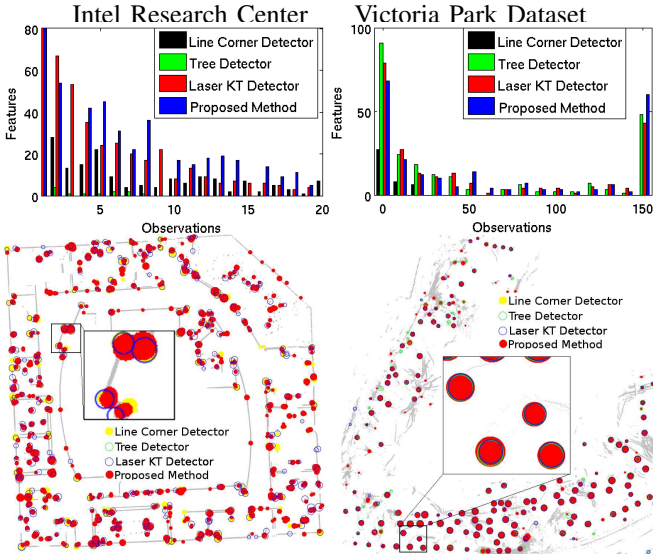


Fig. 5. Repeatability Comparison. Top: histograms show the number of features according to the number of observations of each feature. Bottom: the location of detected features is shown; the size of each marker represents the number of re-observations (large dots denote often-reobserved features). Note: in the histogram for Victoria Park, the peak at 150 observations is the result of the fact that it represents *all* values greater than or equal to 150.

perform comparison [29], [24]. Borrowing ideas from these methods, we use the number of repeated observations and the number of missed detections to evaluate our method.

For our evaluation, we run the proposed method in datasets that contains ground-truthed robot poses. We define two thresholds,  $d_s$  and  $d_b$ , with  $d_s < d_b$ . When a feature is detected, we search for landmarks that it might match. Suppose the closest known landmark is a distance  $d$  from the detected feature; if  $d < d_s$ , we treat the detection as a re-observation of the existing landmark. If  $d > d_b$ , we treat the feature as the first observation of a *new* landmark. For intermediate values, we discard the observation as ambiguous. This simple scheme was selected precisely for its simplicity and reproducibility.

## IV. RESULTS

### A. Repeatability

We measure the repeatability of our feature detector on two well-known datasets: the Intel Research Center dataset (indoor and rectilinear) and the Victoria Park dataset (outdoor with trees and clutter). These two datasets are generally considered to be representative of indoor and outdoor environments, respectively. For these datasets, we used posterior position estimates produced by conventional SLAM methods; therefore, we can easily determine the number of times that a feature should have been observed, and the number of times that it was observed.

In the evaluation method discussed in the previous section, there are two threshold,  $d_s$  and  $d_b$ . In the Intel dataset, we set  $d_s = 0.15m$  and  $d_b = 0.3m$ , and in the Victoria dataset,  $d_s = 1.5m$  and  $d_b = 3m$ . In addition to the proposed feature detector, we provide three comparison detectors (all of which used the same data association procedure):

- Line/corner detector: Line segments are extracted from nearby points using an agglomerative method. If the

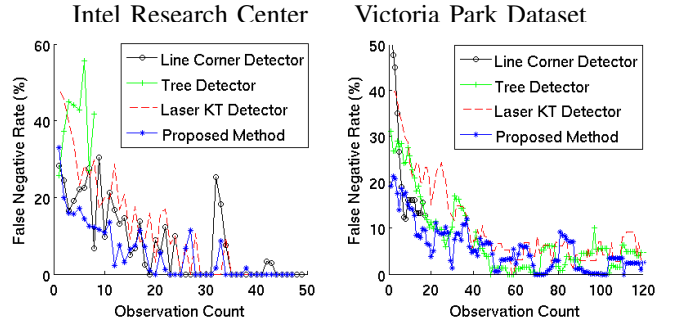


Fig. 6. False Negative Rates Comparison. The plots show the frequency with which a landmark is not correctly detected as a function of how often the landmark is detected. It illustrates that our method achieves low false-negative rates. There is also a clear trend: the more often a landmark is observed, the lower the false positive rate. The very low false negative rates achieved would allow systems to make use of negative information.

endpoints of two lines lie within 1.2 m of each other, and the angle of between the two lines is between 75 and 105 degrees, a corner is reported. The particular method is adapted from [13].

- Tree detector: The standard method of tracking features in Victoria park is using a hand-tuned tree detector with hand-tuned parameters [15]. We used this detector with no additional modifications.
- Laser KT detector: A general-purpose feature detector for LIDAR data with parameters suggested in [20].

As shown in Fig. 5, the performance of the proposed method is generally as good or better than the other three detectors. Although the proposed method extracts fewer features than the Laser KT detector, it has better repeatability, which indicates the proposed method is more robust to noise.

### B. False Negative Rate

False negative rates defined as  $o/(f+o)$  are shown in Fig. 6, where  $f$  denotes the number of re-observation times and  $o$  denotes the number of missing times. As shown in the figure, our detector is able to identify features which are highly stable: they are observed many times with lower false negative rates. The figure also shows that some features are detected but are observed less frequently and less consistently; empirically, these might result from the fact that features do not have equal opportunities to be observed in the two datasets.

### C. Computation Complexity

The comparison to the four feature detectors in CPU time are shown in Table 7. The experimental platform is a laptop with a 2.53GHz CPU, and the programming language is Java. As the data shows, the method runs in real-time.

## V. CONCLUSION

We have described a feature detector for LIDAR data that is capable of detecting stable and repeatable features from virtually any environments. Our method builds upon techniques originally developed for computer vision, but adds a principled noise filtering step and a scale space searching strategy that produces precise feature detections even at large scales.

	Corner Detector		Tree Detector		Laser KT Detector		Proposed Method	
	Victoria	Intel	Victoria	Intel	Victoria	Intel	Victoria	Intel
Total Number of Feature Detections	409	1819	31615	193	57545	8341	49920	7091
Total Time (ms)	566.02	2780.73	193.31	79.21	8150.13	40510.14	2250.13	9401.04
Max Single Step Time (ms)	5.15	10.21	1.72	0.68	100.02	289.22	17.11	48.21

Fig. 7. Comparison of four feature detection methods. The performance of two standard detection methods, our previous work, and our proposed work are shown above. The proposed method detects a wealth of features in both environments, and is significantly faster than our previous method. As shown in Fig 5, these feature detections are highly repeatable.

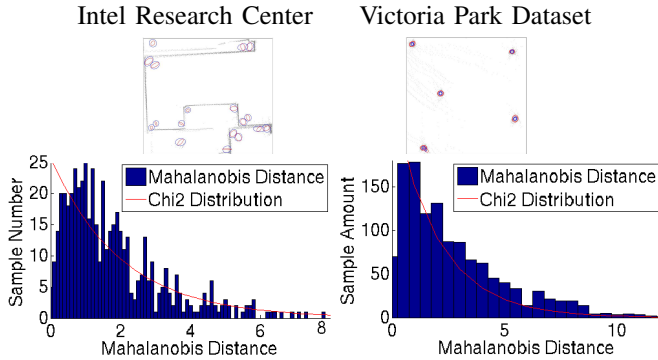


Fig. 8. Comparison between sample covariances and the averaged covariance estimations. Up: red ellipses denote sample covariances and blue ellipses denote averaged covariance estimations. The covariance estimations are conservative in the sense that it slightly overestimate the uncertainty. Bottom: fits  $\chi^2$  distributions to Mahalanobis distances calculated with estimated uncertainties. Red curves denote  $\chi^2$  distributions and blue histograms denote the distribution of Mahalanobis distances. The fitting shows uncertainties estimated in our method are reasonable estimations to real feature uncertainties.

We demonstrated the general-purpose applicability of the method on benchmark indoor and outdoor datasets, where it matches or exceeds the performance of earlier works. We also showed that the method produces reasonable uncertainty estimates, a critical factor for SLAM applications.

#### ACKNOWLEDGEMENTS

This work was supported by U.S. DoD grant W56HZV-04-2-0001 and China NSF International Cooperation grant 60910005.

#### REFERENCES

- [1] F. Dellaert, "Square root SAM," in *Proceedings of Robotics: Science and Systems (RSS)*, Cambridge, USA, June 2005.
- [2] E. Olson, J. Leonard, and S. Teller, "Fast iterative optimization of pose graphs with poor initial estimates," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2006, pp. 2262–2269.
- [3] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent," in *Proceedings of Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, 2007.
- [4] T. Duckett, S. Marsland, and J. Shapiro, "Learning globally consistent maps by relaxation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 4, San Francisco, CA, 2000, pp. 3841–3846.
- [5] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localization and mapping," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 196–207, April 2005.
- [6] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," Carnegie Mellon University, Pittsburgh, PA, Tech. Rep., April 2003.
- [7] R. Eustice, H. Singh, and J. Leonard, "Exactly sparse delayed-state filters," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005, pp. 2428–2435.
- [8] M. R. Walter, R. M. Eustice, and J. J. Leonard, "Exactly sparse extended information filters for feature-based SLAM," *Int. J. Rob. Res.*, vol. 26, no. 4, pp. 335–359, 2007.
- [9] E. Olson, "Real-time correlative scan matching," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, May 2009, pp. 4387–4393.
- [10] P. Nunez, R. Vazquez-Martin, J. del Toro, A. Bandera, and F. Sandoval, "Feature extraction from laser scan data based on curvature estimation for mobile robotics," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, May 2006, pp. 1167–1172.
- [11] G. A. Borges and M.-J. Aldon, "Line extraction in 2d range images for mobile robotics," *J. Intell. Robotics Syst.*, vol. 40, no. 3, pp. 267–297, 2004.
- [12] A. Diosi and L. Kleeman, "Uncertainty of line segments extracted from static pls laser scans," in *SICK PLS laser. In Australasian Conference on Robotics and Automation*, 2003.
- [13] E. Olson, "Robust and efficient robotic mapping," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, June 2008.
- [14] V. Nguyen, S. Gächter, A. Martinelli, N. Tomatis, and R. Siegwart, "A comparison of line extraction algorithms using 2d range data for indoor mobile robotics," *Auton. Robots*, vol. 23, no. 2, pp. 97–111, 2007.
- [15] J. E. Guivant, F. R. Masson, and E. M. Nebot, "Simultaneous localization and map building using natural features and absolute information," *Robotics and Autonomous Systems*, vol. 40, no. 2-3, pp. 79–90, 2002.
- [16] M. Montemerlo, "FastSLAM: A factored solution to the simultaneous localization and mapping problem with unknown data association," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2003.
- [17] Y. Liu and S. Thrun, "Results for outdoor-slam using sparse extended information filters," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002, pp. 1227–1233.
- [18] L. Pedraza, D. Rodriguez-Losada, F. Matia, G. Dissanayake, and J. Miro, "Extending the limits of feature-based slam with b-splines," *Robotics, IEEE Transactions on*, vol. 25, no. 2, pp. 353–366, April 2009.
- [19] R. Zlot and M. Bosse, "Place recognition using keypoint similarities in 2d lidar maps," in *ISER*, 2008, pp. 363–372.
- [20] Y. Li and E. B. Olson, "Extracting general-purpose features from lidar data," in *Robotics and Automation, 2010. Proceedings. ICRA '10. 2010 IEEE International Conference on*, May 2010.
- [21] K. Peterson, J. Ziglar, and P. Rybski, "Fast feature detection and stochastic parameter estimation of road shape using multiple lidar," in *IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems*. IEEE, September 2008.
- [22] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM System Journal*, vol. 4, no. 1, pp. 25–30, 1965.
- [23] C. Tomasi and T. Kanade, "Detection and tracking of point features," Tech. Report CMU-CS-91-132, Carnegie Mellon University, Tech. Rep., April 1991.
- [24] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *In European Conference on Computer Vision*, vol. 1, 2006, pp. 430–443.
- [25] C. Harris and M. Stephens., "A combined corner and edge detection," in *Proceedings of The Fourth Alvey Vision Conference*, 1988, pp. 147–151.
- [26] U. Orguner and F. Gustafsson, "Statistical characteristics of harris corner detector," in *Statistical Signal Processing, 2007. SSP '07. IEEE/SP 14th Workshop on*, Aug. 2007, pp. 571–575.
- [27] F. Mokhtarian and F. Mohanna, "Performance evaluation of corner detectors using consistency and accuracy measures," *Comput. Vis. Image Underst.*, vol. 102, no. 1, pp. 81–94, 2006.
- [28] M. Trajkovic and M. Hedley, "Fast corner detection," *Image and Vision Computing*, vol. 16, no. 2, pp. 75–87, February 1998.
- [29] C. Schmid, R. Mohr, and C. Bauckhage, "Evaluation of interest point detectors," *International Journal of Computer Vision*, vol. 37, no. 2, pp. 151–172, 2000.