Finding Planes in LiDAR Point Clouds for Real-Time Registration

W. Shane Grant*, Randolph C. Voorhies*, and Laurent Itti

***Abstract*— We present a robust plane finding algorithm that when combined with plane-based frame-to-frame registration gives accurate real-time pose estimation. Our plane extraction is capable of handling large and sparse datasets such as those generated from spinning multi-laser sensors such as the Velodyne HDL-32E LiDAR. We test our algorithm on frame-to-frame registration in a closed-loop indoor path comprising 827 successive 3D laser scans (over 57 million points), using no additional information (e.g., odometry, IMU). Our algorithm outperforms, in both accuracy and time, three state-of-the-art methods, based on iterative closest point (ICP), plane-based randomized Hough transform, and planar region growing.**

I. Introduction

Although significant progress has been made in recent years towards 6D simultaneous localization and mapping (SLAM) for point cloud data [1] [2] [3], frame-to-frame registration, a critical step in many SLAM algorithms, remains both a challenge and a significant bottleneck for real-time implementations. The typical approach of using some variant of iterative closest point (ICP) to solve for the transformation between successive frames lacks speed as the number of points increases and lacks accuracy as the density decreases. We propose a 3D plane extraction algorithm, capable of working on large sparse point clouds, that when combined with a plane-to-plane based registration method, provides accurate real-time frame-to-frame registration.

Such registration becomes particularly important when odometry is either inaccurate or unavailable, such as when the sensor is mounted on a bipedal robot or carried by a human. In this paper, we perform registration on data captured from a Velodyne HDL-32E LiDAR sensor, which provides distance measurements 360° horizontally and 40° vertically (using 32 lasers rotating around a vertical axis). Velodyne sensors have marked a turning point in many robotics applications by providing 3D point cloud data at a high refresh rate (10-15 Hz) and long range (50-120 meters). They have been successfully deployed in landmark events such as the DARPA Grand Challenge [4] and Urban Challenge [5], and are now used on a growing variety of autonomous vehicles and robots [6]. However, because the point cloud data generated by these sensors is only dense along the azimuthal direction (figure 1), it has proven difficult to apply standard point cloud registration algorithms, which usually assume uniformly dense data. Hence, thus far, many applications have required that Velodyne data be assisted with odometry to guide frame-to-frame alignment, which is problematic for humanoid robotics. Here, we carry the sensor on a backpack, and we show that our new algorithm can perform frame-to-frame registration with speed and accuracy beyond the state-of-the-art.

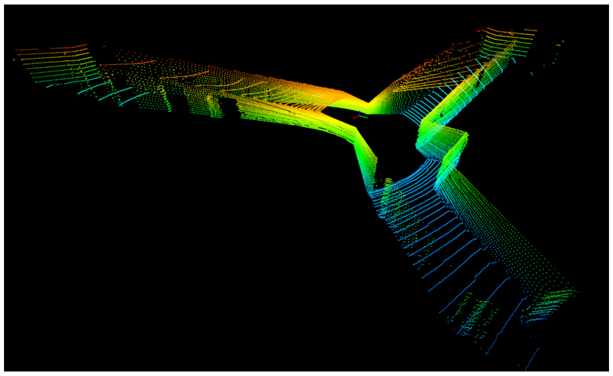


Fig. 1: An example of data captured using a spinning multi-laser sensor (in this case a Velodyne HDL-32E). Note the rapidly increasing sparsity of points as distance from the origin increases.

Our main contribution is to derive a novel Hough-based voting scheme for finding planes in 3D point clouds. Correspondence between extracted planes from frame to frame is then computed using a simple thresholded nearest plane algorithm. The affine transform between two successive sets of corresponding planes is computed using a decoupled optimization, where the translation component is approximated in the least squares sense, and the rotation is determined via Davenport's Q-method. Testing on a large dataset suggests superior performance compared to three state-of-the-art approaches.

II. Previous Work

Substantial previous work has been described on the problems of both frame-to-frame registration as well as 3D plane extraction. Two approaches of particular interest to our discussion are ICP-based methods and plane-to-plane based methods. We describe the challenges which these existing methods face when dealing with data from sensors such as the Velodyne HDL-32E.

A. ICP Approaches

Possibly the most prevalent method for performing frame-to-frame correspondence in point clouds is ICP [7]. Originally designed to register dense point sets between two scans of an object, the ICP algorithm aligns two sets of points iteratively, where each iteration consists of first finding the best

*These authors contributed equally to this work.

The authors are with the Department of Computer Science, University of Southern California, Los Angeles, CA, USA {wgrant, voorhies} at usc.edu

point-to-point correspondences, and then computing a rigid transformation between all corresponding pairs of points. Some variations on point-to-point ICP include point-to-plane [8], which minimizes error along local surface normals, and Generalized ICP [9], which exploits local planar patches in both the source and target scans.

When dealing with large numbers of points, it is common for ICP algorithms to take either a random subsampling of points or to voxelize the data to reduce the computational burden. This can come at a cost of decreased accuracy, as fewer points are used to constrain the transformation between pairs of clouds. There has also been considerable effort to speed up ICP-based approaches through hardware optimization, such as GPGPU acceleration [10] [11] [12], multithreading [13], and sophisticated data structures [14].

Very few ICP-based solutions have been shown to work (without the aid of odometry or other additional guidance) on data acquired from Velodyne-style sensors, where the point clouds are very sparse in at least one dimension. A major issue with point-to-point ICP approaches is that sparse scan-lines on horizontal planes may not reflect the same physical points from one scan to the next. The spacing between scan-lines increases dramatically as distance from origin increases, making it very unlikely that a meaningful point correspondence can be made on these scan-lines from frame to frame. Although the region near the origin is considerably more dense, it is often the case that it does not cover enough physical area to properly constrain registration. With point-to-plane approaches, an additional difficulty lies in accurately and rapidly estimating normals from the sparse data.

B. Plane Approaches

There is an expanding literature on plane finding algorithms for point clouds, which, when coupled with frame-to-frame techniques such as that presented in [15], can provide accurate pose estimation. These plane finding techniques can coarsely be categorized into Hough-based, region growing, or RANSAC-based approaches.

Hough-based methods utilize a buffer in which votes for candidate planes are accumulated. An excellent comparison of many such methods is available in [16]. The randomized Hough transform (RHT) method for plane extraction is one of the most efficient, working iteratively by randomly selecting three points in a cloud, fitting a plane to them, and voting for that plane in the accumulator. When the sum of accumulated votes for a particular plane passes some threshold, all points that lie close to that plane are removed from the cloud, reducing the search space for future iterations. While this approach works well with dense point clouds, as the samples on planes become more sparse, the probability of detecting these planes decreases.

Region growing approaches [17] [18] exploit the structure of the sensor's raw data by working on range images or on rasterized versions of the point cloud. By starting at a given point and expanding outwards (often along scan-lines), plane parameter estimates are updated until adding new points would violate the online estimate. Here again sparsity in one dimension is problematic, especially for rasterization. In addition, these approaches are not directly applicable to Velodyne data because of the underlying assumption that scan-lines on planes produce linear segments, whereas Velodyne scan-lines are conics (figure 2).

Finally, RANSAC [19] based methods either attempt to directly fit planes [20], or merge higher-level analysis to find better fits [21] [22]. Sparseness in one direction is also a challenge for these methods, which additionally suffer in real-time applications because of their high computational cost.

III. Finding Planes

Our plane finding algorithm exploits knowledge about the sensor to first, on a scan-line basis, find and group all points that arise from planar surfaces. Each of these groups of points then casts weighted votes in a Hough accumulator for all possible planes that could explain them. The planes that pass thresholding the accumulator are refined in a final filtering step. We utilize the following notations:

n, $\vec{n}$, $\hat{\vec{n}}$,	a scalar, a vector, a unit vector
M, M^+	a matrix, its Moore-Penrose pseudo-inverse
$\boldsymbol{p}$, $\boldsymbol{p_n}$, $\boldsymbol{p_s}$	a plane, its normal form, its spherical form

Here, $\boldsymbol{p_n} \triangleq \begin{bmatrix} \hat{\vec{n}} & \rho \end{bmatrix}$, where $\hat{\vec{n}}$ is the plane normal, and ρ is distance from the plane to the origin along $\hat{\vec{n}}$. $\boldsymbol{p_s} \triangleq \begin{bmatrix} \theta & \phi & \rho \end{bmatrix}$ where θ is the azimuthal angle in the x-y plane, and ϕ is the inclination angle.

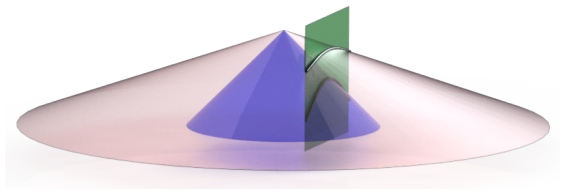
A. The Sensor

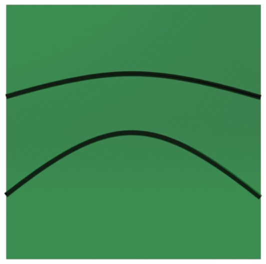
Before discussing how we find points on planar surfaces, it is useful to know the structure of data captured by rotating multi-laser sensors. The Velodyne HDL-32E LiDAR sensor consists of 32 lasers in inclinations ranging from +10.67° to -30.67° (approximately 1.33° angular resolution), which rotate a full 360° (approximately 0.16° angular resolution) at 10 Hz to provide depth measurements from one to seventy meters away. The sensor generates 700,000 points per second, which corresponds to 70,000 points per full revolution, or close to 2,200 points per laser.

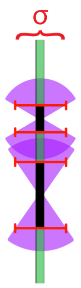
As the sensor head spins, each laser gives a high horizontal density of points, but the spacing between successive laser scan lines causes an increasing sparsity of information as distance from the sensor increases. Although there is also an increase in sparsity in single laser scans as depth increases (adjacent points become farther apart), it is much less severe than that caused by the low vertical angular resolution of the sensor. As mentioned above, this inequality in angular resolution causes many traditional point cloud registration algorithms to work poorly on data generated from rotating multi-laser sensors.

B. Finding Points on Planes

The first step in the plane extraction algorithm is to find all points in the point cloud which originate from planar surfaces. To do this, we look at each laser scan independently and exploit some simple geometry: the intersection of any single rotating laser, which forms a cone in space, with any


(a) Perspective view of laser cones (red and blue) intersecting an actual, physical vertical plane (green).


(b) Frontal view (through plane) of laser scan-lines (black) intersecting the physical plane (green).



(c) Side view of laser intersection (black) through physical plane (green) with error bars (red) for depth measurement. Purple shaded area shows possible plane parameterizations for each laser that could generate the physical plane given the sensor noise.

Fig. 2: The physical design of the sensor causes it to create virtual laser cones as it sweeps each of its lasers through θ (2a). When a plane intersects one of these cones, it generates a conic section of varying curvature depending upon the inclination angle of the sensor and the relative orientation of the plane. (2b). Since the variance of depth measurements, σ, is known, sections of higher curvature lead to more certainty in detected orientation of the plane (2c).

possible plane in the world, forms a conic section (figures 2a and 2b). This conic can be seen by looking at the 1D signal corresponding to depth measurements from an individual laser.

An ideal solution would be to find and fit all possible conics to this signal - conic sections with higher curvature correspond to more confident evidence of planes with a particular orientation, and those with lower or no curvature have much greater uncertainty about the possible planes that generated them (figure 2c). Unfortunately, finding and fitting general conics to unpartitioned data is neither easy nor fast when dealing with thousands of points [23]. As a fast approximation, we look for regions in the signal which correspond to smoothly varying segments that could possibly belong to a conic.

1) Finding Breaks: Approximate conic sections in each laser's 1D signal, $d_\phi(\theta)$, are found by detecting points that break the signal into smooth and non-smooth groups of measurements. This is done using simple filtering operations: a Gaussian kernel smoothes the input signal, and a gradient filter takes several spatial derivatives: $d_\phi(\theta)'$, $d_\phi(\theta)''$, and $d_\phi(\theta)'''$. The Gaussian is parameterized by:

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/2\sigma^2} \tag{1}$$

where we choose $\sigma = 2\,\text{cm}$ to match the variance of our sensor. For computing the gradient we use a simple gradient filter: $\nabla = \begin{bmatrix} -1 & 1 \end{bmatrix}$.

Breakpoints are placed at zero crossings of $d_\phi(\theta)'$ and zero crossings of $d_\phi(\theta)'''$ that have sufficiently high $d_\phi(\theta)''$. Many corners caused by distinct planes feature a nearly smooth transition of depth values, but will be caught as a local extrema in $d_\phi(\theta)'$. Although noisy, zero crossings in $d_\phi(\theta)'''$ can be used to find points where the difference in depth changes abruptly, corresponding to different surfaces in the world. We utilize a threshold parameter, $t_{d''}$, to enforce that only zero crossings of the the third derivative that had a second derivative value greater than $t_{d''}$ are considered break points. This helps compensate for the sensor noise amplified by taking $d_\phi(\theta)'''$. In practice this parameter does not need to be adjusted and we have used a value of $t_{d''} = 0.01$ for all data. As a general rule, we aim to over-segment the signal since under-segmentation leads to overly confident incorrect initial plane estimates.

2) Creating Groups: Once breakpoints are extracted from each laser scan, we place points into groups such that the set of planes to which each group could belong can be calculated. A collection of points between two breakpoints is considered a valid group if it contains at least some minimum number of points, $\#_p$, which we set to 15 (an angular resolution of 2.4°). This constraint enforces a minimum size of planar segments and prevents overly noisy (outlier) points from being considered. This constraint also gives stability to the group centroid, $\vec{c}$, which is used during the voting procedure.

For each group, we find its principal components through the eigenvector decomposition of its points' covariance matrix. From this decomposition, we define the following:

$\hat{\vec{v_3}}$	the principal eigenvector
$\hat{\vec{v_1}}$	the eigenvector corresponding to the smallest eigenvalue
c	the curvature of the group
λ_i	the eigenvalues of the decomposition

$\hat{\vec{v_3}}$ corresponds to the dominant direction of the group of points, which by convention we enforce to point in a clockwise direction relative to the sensor origin. $\hat{\vec{v_1}}$ corresponds to the direction of least variation amongst the points, which we expect to correspond to the plane normal of the actual surface generating these points. However, when the curvature $c \triangleq \frac{\lambda_1+\lambda_2}{\lambda_1+\lambda_2+\lambda_3}$ is low, our uncertainty over $\hat{\vec{v_1}}$ becomes high.

Groups that are close to each other within a small distance threshold t_g and pointing similar directions $(\hat{\vec{v_3}}_i \cdot \hat{\vec{v_3}}_j < t_d)$ are merged and their decomposition is re-evaluated. In the next step, we use these values to vote for plane distributions for each group.

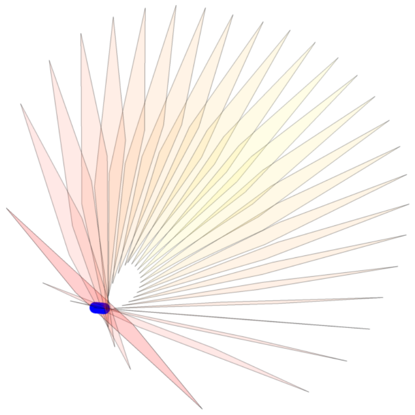


Fig. 3: A set of planes through the blue line (drawn here coming out of the page) as derived from equation 2.

C. Accumulator

To robustly detect planes, we use an accumulator framework similar to that utilized in the randomized Hough transform. During the accumulation step, we represent planes $\boldsymbol{p}$ with their spherical representation $\boldsymbol{p_s}$. This representation is more compact than the $\boldsymbol{p_n}$ form, although this comes at the price of singularities when ϕ is equal to $\pm\frac{\pi}{2}$. These singularities can be handled by ensuring that all possible plane parameterizations, which span the unit sphere, are equally accounted for in the accumulator. This is accomplished by using the ball accumulator of [16], which varies the number of θ bins depending on the angle ϕ such that the bins are uniformly sized. The accumulator is parameterized by the maximum number of bins per dimension, $\#_\theta$, $\#_\phi$, and $\#_\rho$.

D. Voting

For each group in a scan, we need to find a parameterization for the set of all planes that contain that group (see figure 3 for an illustration). This is done by finding all planes that could cause the vector $\vec{v}_3$ centered at the group centroid $\vec{c}$. To ensure that we do not double vote for any particular plane given our accumulator quantization, we need to solve for the θ and ϕ parameters of each plane as we step through the most dense dimension of the accumulator, ρ. To do so, we arrange the following under-constrained linear equation using the $\boldsymbol{p}_n$ parameterization:

$$\underbrace{\begin{bmatrix} c_x & c_y & c_z \\ v_{3x} & v_{3y} & v_{3z} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}}_{\vec{x}} = \underbrace{\begin{bmatrix} -\rho \\ 0 \end{bmatrix}}_{\vec{b}} \tag{2}$$

where $\vec{c}$ is the centroid of the group.

Any solution to this equation will be of the form

$$\vec{n} = \left\{ \alpha\vec{j} + \vec{k} : \vec{j} \in Null(A), A\vec{k} = \vec{b} \right\} \tag{3}$$

and so we solve for $\vec{n}$ by first finding any solution to equation 2 by using the Moore-Penrose pseudo-inverse of A:

$$\vec{k} = A^+ + \left(I - A^+ A\right)\vec{w} \tag{4}$$

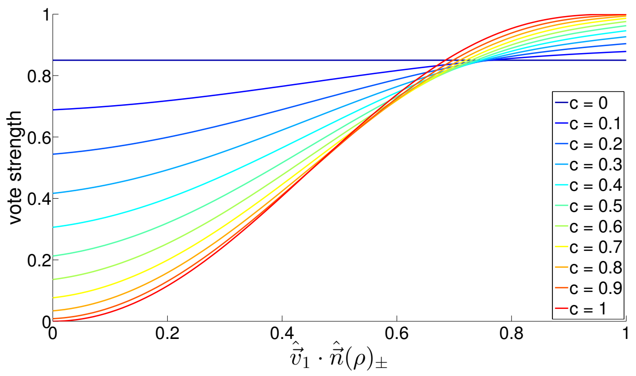


Fig. 4: Voting strength as a function of group curvature and normal similarity. The x axis denotes normal similarity, while the y axis denotes vote strength. The various colored curves represent different levels of curvature.

where $\vec{w}$ is any vector. We need only unit length normals, and so we solve for α with the following quadratic equation:

$$1 = \left\| \vec{j} + \alpha\vec{k} \right\|_2 \tag{5}$$

which gives the following two solutions:

$$\hat{\vec{n}}(\rho)_\pm = \frac{\vec{j}\cdot\vec{k} \pm \sqrt{(\vec{j}\cdot\vec{k})^2 - \left\|\vec{k}\right\|^2 \left(\left\|\vec{j}\right\|^2 - 1\right)}}{\left\|\vec{k}\right\|^2} \tag{6}$$

Once we have two solutions, we find corresponding θ and ϕ by converting $\hat{\vec{n}}(\rho)_\pm$ into spherical coordinates.

We then vote in the accumulator for these two planes, weighting the votes according to group's curvature c, its smallest eigenvector $\hat{\vec{v}}_1$, and the calculated $\hat{\vec{n}}(\rho)_\pm$. When the curvature is high, we trust that the group's $\hat{\vec{v}}_1$ accurately describes the normal of the physical plane generating the group (figure 2); thus we form a weighting function such that in cases where curvature is high, we penalize normals that are not similar to $\hat{\vec{v}}_1$, whereas in cases where curvature is low, we give all possible normals equal weight (figure 4). We model this relationship using a linear combination of a line and a sigmoid (here we use a Kumaraswamy CDF [24] because of its convenient parameterization), where the mixing is defined by the curvature of the segment:

$$\begin{gathered} s = c(1 - (1 - (\hat{\vec{v}}_1 \cdot \hat{\vec{n}}(\rho)_\pm)^2)^3) + \\ (1 - c)(c(\hat{\vec{v}}_1 \cdot \hat{\vec{n}}(\rho)_\pm) + 0.85 - c) \end{gathered} \tag{7}$$

E. Peripheral Planes

When an initial plane detection arises from points belonging to a planar patch that is far away from the sensor and not tangent to any laser, small perturbations in depth give rise to large changes in the estimated normal. This is because of our plane parameterization and the choice to use infinite planes. Since the plane normal vector always connects the origin with the closest point on the infinite plane, small angular

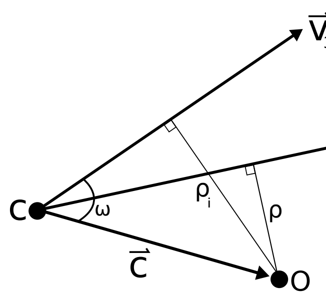


Fig. 5: An illustration of the perturbation of some vector $\vec{v}_3$ to account for noise. O is the sensor origin and C is the centroid of the group. $\vec{v}_3$ is the dominant eigenvector of the group, which generates a normal of length ρ with the origin. Rotating $\vec{v}_3$ about the global Z axis creates a new vector $\vec{v}_{3R}$. We seek this rotation amount given the new desired distance for the normal connecting the origin and $\vec{v}_{3R}$, denoted by ρ_i.

changes in the plane become large distances when occurring at the periphery.

We compensate for this error by making small rotational perturbations to a group's $\vec{v}_3$ vector about the global Z axis up to some maximum amount, $\pm\eta$, dependent on the sensor noise model. Rotating the vector about the Z axis captures most of the variation caused by noisy depth readings (since the variance of LiDAR measurements is usually dominated by the depth measurement, we consider the worst case rotational difference to $\vec{v}_3$ two adjacent point measurements can cause when perturbed in depth).

After each of these perturbations is made, the entire voting process is repeated with the new $\vec{v}_{3R}$. This additional voting, which creates entries in the accumulator around the original vector's entries, makes it more likely groups on the same distant peripheral plane share votes. Without this step small perturbations to groups arising from the same planar patch can cause them to vote for entirely different bins in the accumulator. See figure 5 for an illustration of the perturbation.

Much like during voting, we would like to be able to step through our ρ dimension as we rotate the original $\vec{v}_3$. Given this desired ρ_i, we must solve for the amount of rotation, η_i, that would generate a vector $\vec{v}_3R$ with a normal of length ρ_i. To solve for this, we can take advantage of the definition of the dot product:

$$\cos(\omega) = R_z(\eta_i)\,\hat{\vec{v}}_3 \cdot \hat{\vec{c}} \tag{8}$$

where $R_z(\eta_i)$ is a rotation about the global Z axis by η_i and we have observed that $\vec{v}_{3R} = R_z(\eta_i)\,\vec{v}_3$. $\cos(\omega)$ is known since we know all lengths of the right triangle created by $\vec{c}$, $\vec{v}_{3R}$, and their normal vector (which has length ρ_i).

Simplifying this equation yields:

$$\sin(\eta_i)\alpha + \cos(\eta_i)\beta = \gamma \tag{9}$$

where $\alpha = \hat{c}_y\hat{v}_3x - \hat{c}_x\hat{v}_3y$, $\beta = \hat{c}_x\hat{v}_3x + \hat{c}_y\hat{v}_3y$, and $\gamma = \hat{c}_z\hat{v}_3z$. The $\hat{c}_i$ and $\hat{v}_{3i}$ refer to specific components of $\hat{\vec{c}}$ and $\hat{\vec{v}}_3$. This gives four possible solutions for η_i; if any one of these solutions results in a rotation within the bounds of $\pm\eta$, we perform full voting on the resultant $\vec{v}_3R$. Once we have exceeded $\pm\eta$ by both increasing and decreasing ρ_i, we move on to another group.

F. Filtering and Refitting

Once all groups have voted for their possible planes, we threshold the accumulator by keeping all bins which pass some value, t_a, and discarding the others. This reduces the potential number of planes (depending upon environment - usually on the order of 100,000) to a significantly more manageable amount (on the order of 100 or 1,000). These remaining planes fall into two categories: 1) clusters of good planes, and 2) false positives. The vast majority of these planes will be clusters of good planes, with the size of the clusters depending upon t_a, which is kept as small as possible to permit planes with weak evidence to form clusters. In an ideal case of no noise, these clusters would collapse onto the true planes and the accumulator threshold would be sufficient to filter out false positives. Since real data is noisy, a small amount of empirically derived filtering is necessary to reduce the number of planes down to the correct amount (usually on the order of 10).

We filter these remaining planes to simultaneously combine clusters and remove erroneous detections by performing the following in order:

1) Linearity Filtering: All candidate planes that were voted for by less than two rows are removed. This prevents any single laser from defining a candidate plane and is designed to remove purely erroneous plane detections that survive the accumulator threshold (e.g. consider the same low curvature scan-line as it goes along a corner - there will be more evidence for a false horizontal plane than for either of the vertical planes).

2) Splitting: Individual planes are split apart by finding clusters of groups within the set of all groups that voted for the plane. A cluster is defined by the set of all groups whose $\hat{\vec{v}}_3$ vectors' dot products is greater than zero and whose points are at most t_{split} apart. The former constraint prevents groups from forming on opposite sides of the sensor (keeping in mind our clockwise convention of section III-B.2), while the latter creates distinct planar patches for coplanar surfaces. Once clusters are formed, planes are re-fit to their corresponding points. Clusters that do not meet a minimum number of groups, $\#_{split}$, are discarded. This splitting procedure removes planes with insufficient support and causes the remaining planes to have spatially localized groups.

3) Merging: After splitting apart groups, we perform a merging of similar planes. For each group, we go through all of the planes that it voted for and try to find the maximal mode for the distribution of normals. We then remove any planes that are too far from this mode (normal dot product less than t_{merge}). We next merge planes that are nearby by finding those that voted for the same set of groups. To do so, we formulate a graph with nodes representing each candidate

Parameter	Category	Value
t_g	Creating Groups (III-B.2)	3.0 cm
t_d	Creating Groups (III-B.2)	0.8
$\#_\theta$	Accumulator (III-C)	180
$\#_\phi$	Accumulator (III-C)	90
$\#_\phi$	Accumulator (III-C)	600
η	Peripheral Planes (III-E)	2°
t_a	Filtering (III-F)	1.0
t_{split}	Splitting (III-F.2)	0.9
$\#_{split}$	Splitting (III-F.2)	2
t_{merge}	Merging (III-F.3)	0.9
t_{dist}	Registration (IV)	2.0 cm
t_{dot}	Registration (IV)	0.5

TABLE I: List of parameters (and the locations of their descriptions) used during our evaluation. Although the algorithm has many parameters, in practice these are quite stable for different environments and can be reasoned about intuitively (e.g. dot products thresholds can be though of as similarity ratios).

plane. Two nodes are connected with an edge if any given group voted for both planes. Once the graph is formed, all connected components within it are computed, and all planes in each component are replaced by a single plane fit to the union of all points covered by the groups in that component.

4) Growing: As a final refinement step, we grow regions over the points in each plane outwards until either the distance between consecutive points is too great, or the distance from the point to the plane is too high. Once this new set of points is found, a final least squares fit is performed according to the formulation of [25] to find the final plane parameters, as well as the covariance and Hessian of the plane. Region growing is merely a refinement step that allows us to consider points which were excluded during group formation.

IV. Registration

For plane registration we utilize the frame-to-frame approach described in [25]. This approach decouples the solutions for rotation and translation, and computes them using efficient closed-form solutions. However, because the data rate of our sensor is very high and our relative translation between scans is low, we use a much less complex correspondence solver. To find correspondences between planes, we simply find the planes $\boldsymbol{p_{n_a}}$ and $\boldsymbol{p_{n_b}}$ such that $\|\rho_a\hat{\vec{n}}_a - \rho_b\hat{\vec{n}}_b\|$ is minimized, and the following two constraint equations are satisfied:

$$\hat{\vec{n}}_a \cdot \hat{\vec{n}}_b > t_{dot} \tag{10}$$

$$\frac{|\rho_a - \rho_b|}{(\rho_a + \rho_b)} < t_{dist} \tag{11}$$

which enforce that matched planes are close together and nearly parallel to each other.

V. Evaluation

We evaluate our plane finding algorithm within the context of a pure frame-to-frame registration problem with no odometry or IMU assistance. Plane-to-plane registration requires well estimated planes to achieve good results; not having enough planes to constrain rotation or translation, or having consistent erroneous planes, quickly leads to massive errors in pose estimation. We compare against the Slam6D ICP implementation in 3DTK [26] as well as two other plane extraction algorithms plugged into our plane to plane registration framework: randomized hough transform [16], and a region growing approach [17]. Parameters used during evaluation can be seen in table I.

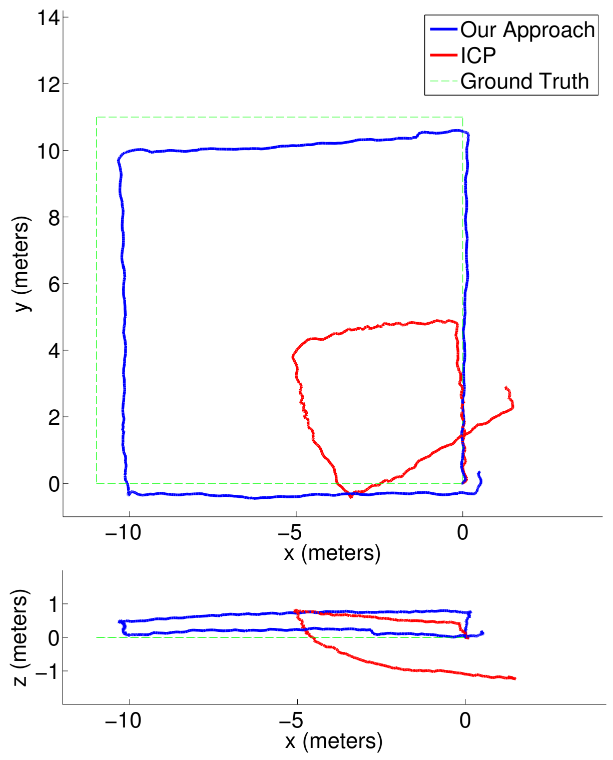
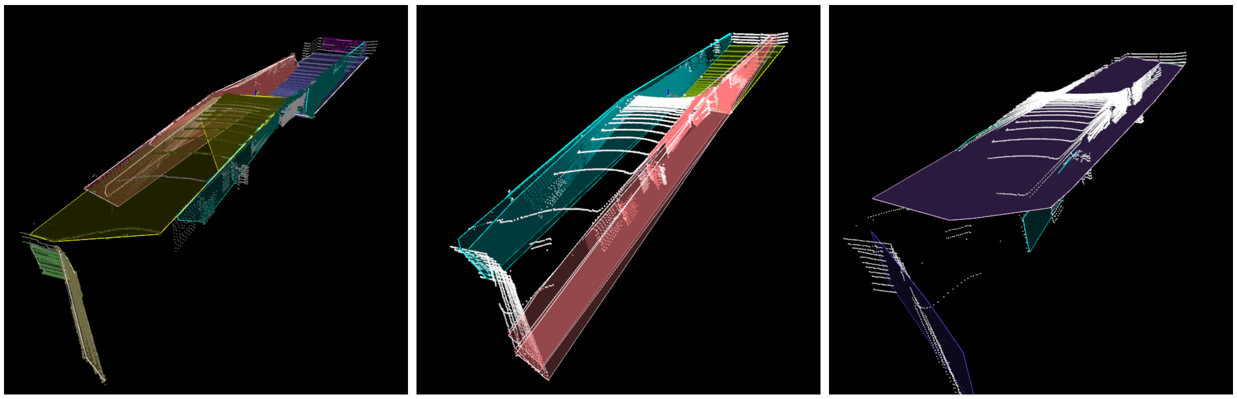


Fig. 6: Paths generated by frame-to-frame registration using our approach (blue) and ICP (red) as the sensor was walked on a backpack rig around an indoor 11m per side square hallway (green).

A. Dataset

We evaluate all algorithms on a dataset collected from an indoor office environment. A perfectly square ($11.1\,\mathrm{m}^2$) closed path through a hallway is taken by a human walking at a normal pace through the environment. Though the sensor is rigidly affixed to the backpack, its high center of mass causes it to sway during movement, exacerbating the motion induced by walking. The sensor is mounted at a 10° tilt, such that the forward viewing angle goes from +0° to -40°, and the rear viewing angle from +40° to -0° in the vertical direction.

Results for the path reconstruction are shown in figure 6. Although the RHT and region growing approaches were occasionally able to achieve accurate frame-to-frame registration over small segments, they had so many catastrophic registration failures that their paths are not shown in the figure. In the case of RHT, these failures occurred due to lack of constraint by missing important planes in front of



(a) Our algorithm (b) Randomized Hough Transform (c) Region Growing

Fig. 7: An example frame from our dataset which shows the weaknesses of two popular plane extraction algorithms compared to our approach. Note that while the Randomized Hough Transform (b) does a good job of detecting the side walls and floors, it misses the ends of the hallway which leads to an under-constrained registration problem. While region growing (c) finds more planes than RHT, it struggles with the disparate sparsity of the data, ultimately leading to poor plane estimation and under-constrained registration.

Method	Frame Time (ms)	Distance Traveled (m)	Final Error (m)
Ideal	100	44	0
Our Method	46	44.228	0.625
ICP	463 / 192	24.072	3.398
Randomized Hough Transform	535 / 86	85.631	5.461
Region Growing	157	86.319	5.025

TABLE II: Evaluation metrics from the discussed data set. For each method, we show the average frame time, the total distance traveled, and the final error between the start point and end point. For reference, the first row shows the ideal values for each metric. Note that both ICP and the Randomized Hough Transform rows have two frame times listed. The first ICP time was recorded using a single thread, while the second was recorded while using all 12 cores of the test machine. The first Randomized Hough Transform time was recorded with the stock algorithm from 3DTK [26], while the second version uses a custom vectorized random number generator.

and behind the sensor, at the far ends of hallways (figure 7b). In the case of region growing, the approach found many erroneous planes due to its inability to cope with the conic nature of scan-lines captured by the sensor (figure 7c). Our plane finding algorithm produced no false planes on this dataset and missed sufficiently few planes that accurate path reconstruction was achieved (figure 7a).

ICP had fairly good rotation estimation largely due to the high horizontal density of points collected from our sensor. However, the increasing sparsity of scan-lines as distance from the origin increases gave ICP some trouble estimating accurate translations. The overall drift in rotation for ICP is quite small, but the translation is off by a significant amount.

The overall results for all algorithms is presented in table II. Timing was evaluated on an Intel Core i7 X990 with 12Gb of memory. All algorithms were single-threaded unless denoted otherwise.

VI. Discussion and Conclusion

The proposed approach was developed to directly exploit the particular structure present in point clouds generated by rotating multi-laser sensors. As mentioned in the introduction, these sensors have become very popular in recent years, thanks to their ability to capture the entire 3D surroundings of a robot at a high frame rate. However, until now robust and efficient algorithms for frame-to-frame registration without assistance of odometry or other sensors has not been available. We believe that our success in designing and validating a plane-based approach that exploits rather than suffers from the geometry of the sensor opens promising new research and application opportunities in robotics.

In testing, we found that the algorithm was successful in detecting many small planar regions accurately and reliably. This is critical for frame-to-frame registration. Future extensions of this algorithm include integrating it into a real-time SLAM framework. This would consider each planar patch found by our algorithm as probabilistic evidence for a real physical structure in the world, which should help address any merging not handled by the filtering, as the SLAM algorithm progressively converges towards a coherent model of the world. Another interesting future direction is to explicitly consider the degenerate cases when the frame-to-frame matching becomes under-constrained, for example by triggering a helper custom ICP-based matcher along the degenerate dimensions in these situations. This could help the algorithm work in less structured environments.

Future evaluations will explore outdoor structured environments, though this will require a more rigorous ground truth scheme. Initial testing in such environments has been very encouraging.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support from the Defense Advanced Research Projects Agency (DARPA DRC: N65236-12-C-3884), the National Science Foundation (CMMI-1235539), the Army Research Office (W911NF-11-1-0046), and U.S. Army (W81XWH-10-2-0076). The authors affirm that the views expressed herein are solely their own, and do not represent the views of the United States government or any agency thereof.

The authors would also like to thank John Shen and Jimmy Tanner for their suggestions while deriving the voting equations in this work.

REFERENCES

[1] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*. Acapulco, Mexico: IJCAI, 2003.

[2] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[3] M. Kaess, A. Ranganathan, and F. Dellaert, "isam: Incremental smoothing and mapping," *Robotics, IEEE Transactions on*, vol. 24, no. 6, pp. 1365–1378, 2008.

[4] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, "Stanley: The robot that won the darpa grand challenge," *The 2005 DARPA Grand Challenge*, pp. 1–43, 2007.

[5] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[6] E. Guizzo, "How google's self-driving car works," Oct. 2011. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works

[7] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[8] C. Yang and G. Medioni, "Object modelling by registration of multiple range images," *Image and vision computing*, vol. 10, no. 3, pp. 145–155, 1992.

[9] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp," in *Proc. of Robotics: Science and Systems (RSS)*, vol. 25, 2009, pp. 26–27.

[10] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*. IEEE, 2011, pp. 127–136.

[11] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, *et al.*, "Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 2011, pp. 559–568.

[12] D. Qiu, S. May, and A. Nüchter, "Gpu-accelerated nearest neighbor search for 3d registration," *Computer Vision Systems*, pp. 194–203, 2009.

[13] A. Nüchter, "Parallelization of scan matching for robotic 3d mapping," in *Proceedings of the 3rd European Conference on Mobile Robots (September 2007)*. Citeseer, 2007.

[14] J. Elseberg, D. Borrmann, and A. Nuchter, "Efficient processing of large 3d point clouds," in *Information, Communication and Automation Technologies (ICAT), 2011 XXIII International Symposium on*. IEEE, 2011, pp. 1–7.

[15] K. Pathak, A. Birk, N. Vaskevicius, M. Pfingsthorn, S. Schwertfeger, and J. Poppinga, "Online three-dimensional slam by registration of large planar surface segments and closed-form pose-graph relaxation," *Journal of Field Robotics*, vol. 27, no. 1, pp. 52–84, 2009.

[16] D. Borrmann, J. Elseberg, K. Lingemann, and A. Nüchter, "The 3d hough transform for plane detection in point clouds: A review and a new accumulator design," *3D Research*, vol. 2, no. 2, pp. 1–13, 2011.

[17] K. Georgiev, R. T. Creed, and R. Lakaemper, "Fast plane extraction in 3d range data based on line segments," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, Sept., pp. 3808–3815.

[18] G. Vosselman, B. G. Gorte, G. Sithole, and T. Rabbani, "Recognising structure in laser scanner point clouds," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 46, no. 8, pp. 33–38, 2004.

[19] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[20] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–4.

[21] S.-Y. An, L.-K. Lee, and S.-Y. Oh, "Fast incremental 3d plane extraction from a collection of 2d line segments for 3d mapping," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, Oct., pp. 4530–4537.

[22] O. Gallo, R. Manduchi, and A. Rafii, "Cc-ransac: Fitting planes in the presence of multiple surfaces in range data," *Pattern Recognition Letters*, vol. 32, no. 3, pp. 403–410, 2011.

[23] X. Yang, "Curve fitting and fairing using conic splines," *Computer-Aided Design*, vol. 36, no. 5, pp. 461 – 472, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010448503001192

[24] "A generalized probability density function for double-bounded random processes," *Journal of Hydrology*, vol. 46, no. 12, pp. 79 – 88, 1980. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0022169480900360

[25] K. Pathak, A. Birk, N. Vaskevicius, and J. Poppinga, "Fast registration based on noisy planes with unknown correspondences for 3-d mapping," *Robotics, IEEE Transactions on*, vol. 26, no. 3, pp. 424–441, June.

[26] A. G. J. U. Bremen) and K.-B. S. G. U. of Osnabrck), "3dtk - the 3d toolkit," Mar. 2012. [Online]. Available: http://slam6d.sourceforge.net