Mitigating Shadows in Lidar Scan Matching using Spherical Voxels

Matthew McDermott and Jason Rife[2]

***Abstract*— In this paper we propose an approach to mitigate shadowing errors in Lidar scan matching, by introducing a preprocessing step based on spherical gridding. Because the grid aligns with the Lidar beam, it is relatively easy to eliminate shadow edges which cause systematic errors in Lidar scan matching. As we show through simulation, our proposed algorithm provides better results than ground-plane removal, the most common existing strategy for shadow mitigation. Unlike ground plane removal, our method applies to arbitrary terrains (e.g. shadows on urban walls, shadows in hilly terrain) while retaining key Lidar points on the ground that are critical for estimating changes in height, pitch, and roll. Our preprocessing algorithm can be used with a range of scan-matching methods; however, for voxel-based scan matching methods, it provides additional benefits by reducing computation costs and more evenly distributing Lidar points among voxels.**

I. Introduction

Lidar Odometry is a dead reckoning technique in which sequences of Lidar range scans are registered on top of each other to estimate vehicle movement. In contrast with sensing modalities like GNSS and INS, the accuracy of a Lidar scan match (and thus the resulting odometry estimate) is a strong function of both sensor quality and the geometric properties of the Lidar's immediate surroundings. In this paper we propose a new approach to mitigate shadows, which occur when regions of the surroundings are occluded (shadowed) by a closer object. The edges of the shadowed regions move when the Lidar moves. Moving shadows violate the assumption that the surroundings are static and, thereby, cause systematic biases in the scan-matching process.

Broadly speaking, existing scan-matching methods fall into one of three categories: end-to-end machine-learning (ML) methods that directly relate two point clouds, feature-based methods that classify groups of points as geometric structures recognizable across point clouds, and voxel-based methods that cluster points via a grid to enable alignment between clouds. In concept, our shadow removal method can be applied as a pre-processing step for any of these methodologies. As such, it is instructive to provide a brief overview of each category.

End-to-end ML methods estimate transformations directly from a pair of raw point clouds, either through direct estimation [1] or by projecting the 3D point cloud to a 2D *spin image* and extracting contextual information from the scene [2]. ML methods are advantageous in that they have the potential to achieve high accuracy because they can capture subtle effects in real-world data not easily captured by simple geometric models. However, ML-based methods are potentially limited because of their unpredictable patterns of error [3] and because of persistent challenges in registering point clouds that are only partially overlapping [4]. Applying shadow-removal as a preprocessing step may help reduce the complexity of the ML-based model required to compute the transformation between two scans.

Feature-based scan-matching methods identify structures in the point cloud and match those structures between scans. The simplest such approach, dubbed *Iterative Closest Point* (ICP), simply matches each individual point between two scans [5]. Early ICP algorithms are quite computationally intensive because of the large number of possible correspondences, so ICP variants have been developed that accelerate processing by limiting correspondence to a local neighborhood [6], by down-sampling the number of points used in each scan [7, 8], or by improving initialization [9, 10]. Other feature-based methods identify and correspond edges [11, 12], planes [13], or regions of similar Lidar reflectivity [14] between scans. Mixed methods also exist that extract then match features using an ML-based approach, such as a deep neural-network [15]. Shadow removal has the potential to aid feature-based methods by ensuring that shadow edges are not labeled as valid features.

Voxel-based methods subdivide the scene into a grid of adjacent volume elements (aka, *voxels*). The distribution of points within each voxel is characterized, and scans are aligned to maximize the similarity of the distributions within each voxel. The best known voxel-based method is the *Normal Distributions Transform* (NDT), which describes the distribution of points in each voxel using a Gaussian density function [16]. Unlike feature-based methods, NDT does not ascribe any specific shape to characterize surfaces, a property which makes it more robust to processing scenes with arbitrary terrain [17]. NDT is also known for being computationally efficient [17]. Another recently introduced

Matthew McDermott is a student in the Mechanical Engineering Ph.D. program at Tufts University in Medford, MA. He works in the Automated Systems and Robotics Laboratory (ASAR) with Dr. Jason Rife. He received his B.S. and M.S. degrees in Mechanical Engineering at Tufts University, matthew.mcdermott@tufts.edu

[2]Jason Rife is a Professor and Chair of the Department of Mechanical Engineering at Tufts University in Medford, Massachusetts. He directs the Automated Systems and Robotics Laboratory (ASAR), which applies theory and experiment to characterize integrity of autonomous vehicle systems. He received his B.S. in Mechanical and Aerospace Engineering from Cornell University and his M.S. and Ph.D. degrees in Mechanical Engineering from Stanford University. jason.rife@tufts.edu

voxel-based approach is the *Iterative Closest Ellipsoidal Transformation* (ICET), a method notable for analytically predicting solution accuracy as a function of scene geometry [18, 19]. ICET also includes a capability to recognize and exclude point groupings compromised by *Aperture Ambiguity* [20]. Shadow removal can benefit voxel-based scan matching by mitigating systematic errors that corrupt voxel point distributions. As we will show, shadow mitigation can also reformulate the voxel grid for more efficient and accurate computations.

The key contributions of this paper are to introduce a novel shadow-removal method, which acts as a preprocessing step before lidar scan-matching, and to characterize the benefits of the shadow-removal method via simulation. In our performance evaluation, we focus on voxel-based scan matching, and in particular on ICET. Because ICET predicts the error covariance of the output states, it is possible to demonstrate clearly that shadow mitigation is effective if simulations show the predicted accuracy matches the true accuracy after shadow mitigation is applied (but not before). In the following sections, we will explain the shadowing error mechanism, detail our shadow-removal approach, and evaluate its performance using high-fidelity simulations.

II. Range Shadowing Effect

To help explain the impact of range shadowing on voxel-based scan matching, this section provides an illustrative geometric model. Consider the scenario shown in Fig. 1. The Lidar moves with a vehicle through a distance from one position to another (from the cross labeled 1 to the cross labeled 2). The Lidar beam intersects a column (shown as a circular cross-section in the figure) at a distance ρ_L from the Lidar. The column casts a shadow, which extends through many voxels including a voxel of interest (dashed rectangle). The voxel is a distance of approximately ρ_V from the column.

As the Lidar moves, the edge of the shadow shifts by a distance δ (which we measure parallel to). The Lidar beam and shadow edge are tangent to the column at a point that moves little in response to Lidar movement, assuming the radius of the column is small compared to ρ_L. For this reasonable case, we can use similar triangles to give:

$$\delta = \frac{\rho_v}{\rho_l} \tag{1}$$

As the Lidar moves (toward the top of the page), the edge of the shadow within the voxel moves in the opposite direction, toward the bottom of the figure. This decreases the size of the shadowed region and increases the width of the point cloud in the voxel. Algorithms like NDT and ICET track the mean location of points within the voxel, so as the shadow recedes, the mean shifts down by $|\Delta\bar{x}|$. Using a one-dimensional, uniform-density approximation for the first moment, the change in the mean is:

$$|\Delta\bar{x}| = \frac{1}{2}\delta \tag{2}$$

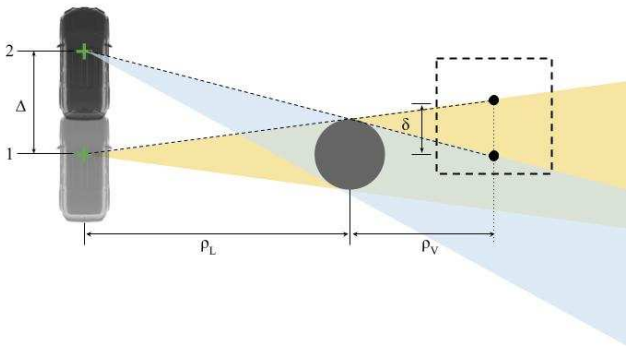


Fig. 1: Schematic of shadows created by a moving Lidar (Vehicle graphic credit: ShapeNet [21])

Combining (1) and (2) gives:

$$|\Delta\bar{x}| = \frac{\rho_v}{\rho_l}\frac{}{2} \tag{3}$$

According to this equation, the point cloud has an apparent shift $|\Delta\bar{x}|$ equal to half , scaled by a distance ratio. The distance-ratio is $\rho_v/\rho_l \in [0, \infty)$, which is zero just behind the column and increasingly large farther behind it. (We assume the Lidar motion is small enough that the shadow edge always crosses through the voxel.)

From our analysis of Fig. 1, we conclude importantly that the shadow (i) impacts many voxels in which it (ii) amplifies apparent vehicle motion. The amplification of vehicle motion occurs because, from the point of view of the moving Lidar, the column appears to shift downward (on the page) while the shadow appears to shift even farther downward. Since the shadow edge cuts through many voxels, a systematic bias is introduced into the perceived motion inferred from all of those voxels. This systematic bias affects all voxels along the edge of the shadow, and according to (3), the bias increases moving progressively farther away from the column.

In the literature, we observe two mechanisms that have been proposed for mitigating the effects of shadowing. The first approach is to remove all voxels on or near the ground plane [12, 22]. This approach is a widely-used heuristic, but it comes with a cost. Removing the ground plane signficantly reduces the ability of Lidar to estimate pitch, roll, and vertical translation. Moreover, the heuristic assumes that shadows lie predominantly on a flat plane (e.g. the road), an assumption that may not be valid in some urban areas (when Lidar shadows are cast on walls) or in offroad terrain.

The second approach looks for jumps in a 1D ranging signal [23]. These jumps correspond to adjacent lidar measurements that reflect from different surfaces (e.g. the column or the ground behind it) and aid in detecting the edge of a shadow. Because these methods only apply to 1D line scans, our new approach, introduced in the following section, adapts the jump-detection concept to apply to 2D scans (e.g., sampled over both azimuth and elevation).

III. Voxel-based Jump Detection

In this section, we apply a spherical-gridding approach in a novel way, to mitigate Lidar shadows. Our approach operates before each scan match, excluding problematic points from the primary scan and the secondary scan, which is transformed to align with the primary.

A. Primary Scan

It is straightforward to convert a points from a Cartesian vector $\mathbf{q}$ to a spherical form with radius r, azimuth , and elevation β. They are related by:

$$\mathbf{q} = \begin{bmatrix} r\cos(\ )\cos(\beta) \\ r\sin(\ )\cos(\beta) \\ r\sin(\beta) \end{bmatrix} \tag{4}$$

If the Lidar unit is modeled as a point, the spherical description is useful because each beam emanates radially outward along a particular azimuth and elevation β. In a spherical grid, Lidar beams do not cross grid boundaries, so it is easy to recognize the nearest radial object and exclude any objects behind it, which are likely to be shadowed. By extension, it is not necessary to fully populate the spherical grid with voxels in the radial direction, since we need only consider the voxel(s) associated with the nearest object. In fact, it is convenient to define only one radial voxel for each azimuth and elevation direction, where the radial limits are adapted to fit the nearest object. This concept for a voxel with adaptive radial boundaries is illustrated in Fig. 2; an algorithm for obtaining the adaptive boundaries is described below and summarized as Algorithm 1.

In constructing our voxel grid, we start with a set of wedge-shaped voxels (no radial limits) indexed by lower azimuth and elevation limits $_i$ and β_j, respectively. If the Lidar points in scan K are indexed $k \in K$, then each point is characterized by the coordinates $^{k)}r$, $^{k)}$, and $^{k)}\beta$. For voxel (i, j), define the set of radial coordinates to be $V_{i,j}$:

$$\begin{aligned} V_{i,j} = \{\ ^{k)}r\} \quad \forall k \quad \text{s.t.} \\ ^{k)} \in [\ _i, \alpha_{i+1}) \\ ^{k)}\beta \in [\beta_j, \beta_{j+1}) \end{aligned} \tag{5}$$

Our approach to segmenting the nearest object is to add an outer radial limit $\bar{r}_{i,j}$ if a radial jump between scan points is detected. This generalizes the 1D jump-detection methods of [23] to a spherical grid. An inner radial bound $\underline{r}_{i,j}$ is also introduced, to screen out low-density stray points that may appear near the Lidar unit. Note that the algorithm depends on two parameters, the jump-distance threshold T and the minimum number of points N that define a valid object.

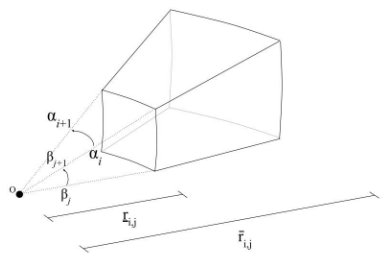


Fig. 2: Voxel boundaries

B. Secondary Scan

Scan matching methods compute a rigid transformation that aligns (or *registers*) a secondary scan to the primary. In general, the shadow-mitigation process of Algorithm 1 is intended as a preprocessing step for both scans. In voxel-based methods, however, the benefits of shadow-matching can be attained by applying the algorithm only to the primary scan. Since a common voxel grid is used for both scans, since points for either scan are only analyzed if they fall into a voxels, and since the voxels exclude shadowed regions of the scene, then the shadow-mitigation benefits established for the first scan transfer to the second through their shared voxelization. This simplification gives a slight efficiency benefit with no apparent impact on accuracy, so we adopt it for our simulations described below.

C. Iterative Scan Matching

Once shadow mitigation has been applied, the rigid transformation relating the scans can be estimated using existing scan matching techniques. For compatibility with existing algorithms, we frame the rigid transformation in Cartesian coordinates, in terms of a rotation matrix and a Cartesian translation through the scalars $\{x, y, z\}$. For each point k in the secondary scan, the rotation and translation can be applied to map the second-scan location $^{k)}\mathbf{p}$ to its equivalent location $^{k)}\mathbf{q}$ in primary-scan coordinates:

$$^{k)}\mathbf{q} = \mathbf{R}\ ^{k)}\mathbf{p} \qquad x \quad y \quad z\Big]^T \tag{6}$$

Algorithm 1 Adaptive Radial Boundaries

```
Initialize point cloud K
Transform all points k to spherical coordinates with (4)
for each wedge-shaped voxel (i, j) do
    Extract radii to obtain V_{i,j} with (5)
    Sort V_{i,j} (ascending) and assign indices l ∈ [0, L]
    Set default index of inner bound to l_min = 0
    Set default index of outer bound to l_max = L
    for each ordered point l (after point 0) do
        compute difference e_l = r_{i,j}(l)    r_{i,j}(l    1)
        if jump detected (e_l > T) then
            if sufficient points (l    l_min > N) then
                reset outer bound to l_max = l    1
            else
                reset inner bound to l_min = l
            end if
        end if
    end for
    Set r_{i,j} = r_{i,j}(l_min) and r̄_{i,j} = r_{i,j}(l_max)
    Exclude all points with l < l_min or l > l_max
end for
```

For voxel-based scan-matching methods like NDT and ICET, Algorithm 1 both eliminates shadowed regions of the scene and defines a useful voxel grid. Any points from the primary or secondary scan whose locations ${}^{k)}\mathbf{q}$ fall outside the defined voxels are excluded from analysis. Point distributions are then compared for each voxel that contains a sufficient number of points from both scans, in order to compute the rotation matrix $\mathbf{R}$ and the translation $\{x, y, z\}$.

IV. Illustrative Example

In order to better understand the proposed shadow-mitigation algorithm, it is helpful to explore an example in detail. Consider the simulated scene shown in Fig. 3, where a vehicle using Lidar for navigation (the *ego-vehicle*) passes a series of round columns that occlude portions of the ground plane and wall. The figure shows four views of the same scene. A visual rendering of the vehicle and scene are shown in Fig. 3a. Simulated Lidar scan points observed as the vehicle passes the last two columns are shown in Fig. 3b; the same image also shows four wedge-shaped bins (in selected directions). The voxels generated by Algorithm 1 are shown in Fig. 3c; the same image also shows the covariance ellipsoids describing the distribution of points in those voxels. In moving from Fig. 3b to Fig. 3c, it is clear that Algorithm 1 prunes voxels in some directions. In order to understand the pruning process (and the process of determining the inner and outer radial bounds for each voxel), Fig. 3d visualizes the point distributions in the four wedges identified in Fig. 3b.

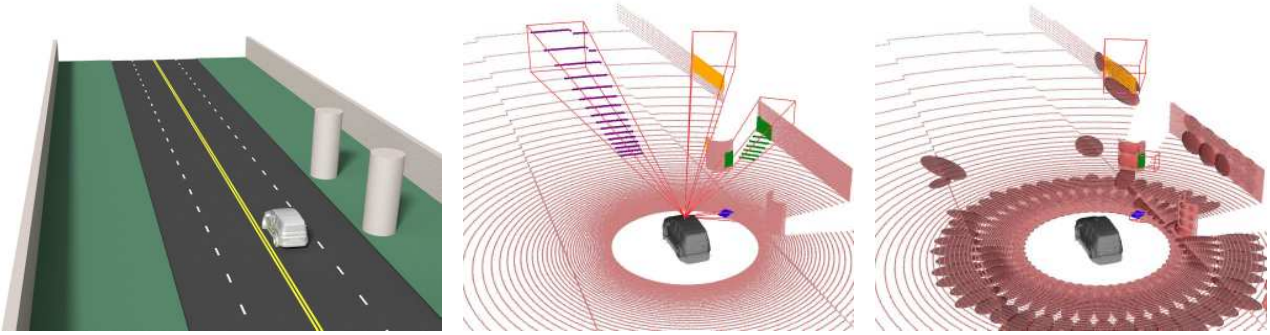
The four wedges were selected to highlight different aspects of how Algorithm 1 functions. The wedges are labeled *case a* through *case d*. In *case a*, marked with purple, the points lie on parallel scan lines crossing the ground plane. Because they are separated by equal intervals in elevation angle, scan lines are separated by an increasing distance along the ground moving toward the horizon. The scan lines are never close enough to accumulate N points without a radial distance jump larger than the threshold T. As a consequence, all of the points in the wedge are excluded, so no associated voxel appears in Fig. 3c. In *case b*, marked with yellow, a very small cluster of points occurs at a short distance (about 12 m) and a much larger cluster of points begins at a farther distance (about 25 m). The nearer points are the edge of a column; the farther points are returns from an oblique section of the wall. There are too few points in the near cluster (less than N), so the inner and outer radial bounds are determined based on the farther cluster, with bounds indicated by the black box shown for *case b* in Fig. 3b. The same yellow-coded voxel appears in 3D in Fig. 3c. By contrast, *case c*, marked with green, captures the other edge of the same column, but with more than N points in the nearer cluster. As a consequence, the radial bounds of the voxel bracket the column and exclude the wall behind, as shown by the black box for *case c* in Fig. 3b and by the green-coded voxel in Fig. 3c. Finally, *case d*, marked by blue, captures a section of the ground plane near to the ego vehicle. All of the blue points in the wedge shown in Fig. 3b and in Fig. 3b are close together, and so no points are excluded from the voxel. The inner radial bound $\underline{r}$ and outer bound $\bar{r}$ capture all points in the wedge, as shown by the black box for *case d* in Fig. 3b.

In this illustration, and in our subsequent performance evaluations, the bins are uniformly 7.2 wide in azimuth and elevation, the distance threshold is $T = 0.2$ m, and the minimum point count is $N = 50$. The four cases described above illustrate the tradeoffs in setting these parameters. The angular scan density is higher in the azimuth direction than the elevation direction, so setting the threshold T is non-trivial. The selected threshold (0.2 m) is low enough to clearly distinguish most occluded objects (e.g. the wall and the column behind) and high enough to cluster points on most oblique surfaces (e.g. the ground plane near the ego vehicle and the vertical wall at the edge of the road); however, the threshold is not high enough to link points on a highly oblique and distant horizontal surface (e.g. the ground plane far from the ego vehicle). Increasing T would allow us to capture more of the ground plane, but at the expense of losing the ability to exclude some shadows.

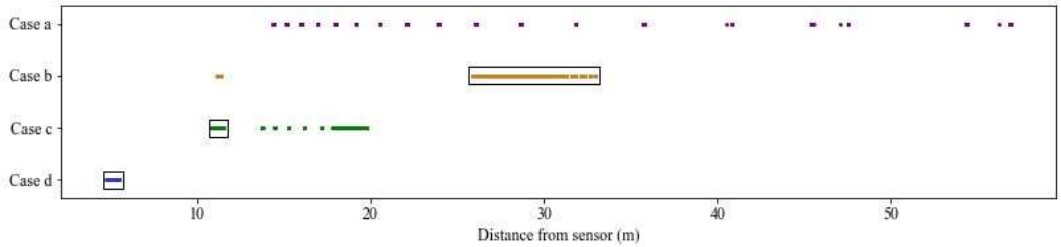
Note the minimum point count N was intentionally set to be higher than the number of points spanning a wedge along a single line scan (at fixed elevation for a rotating Lidar). This means that a voxel is defined for a surface only if least two scan lines spaced closer than T cross the wedge. If N were lower, then a single scan line would be selected to define the voxel in *case a*, which would result in malformed distribution ellipsoid (as compared to those shown in Fig. 3c, where the distribution of points in ground-plane voxels is represented by a well-defined flat, or *oblate*, ellipsoid). At the same time, N cannot be arbitrarily high, as the edge of a nearby object might be excluded, thus failing to remove a shadow. As is, the nearby column edge in *case b* is not recognized as an occluding surface, even though the column casts a shadow on the wall behind. Arguably, *case b* represents a rare misclassification. In practice, we have observed our value for N to be suitable, such that surfaces with fewer than N points did not introduce a strong enough shadow to observably bias performance, as evidenced by the results of the next section.

An additional important consideration is that it is helpful to pad the radial limits slightly, to increase the size of each voxel. The reason is that surfaces seen in the secondary image may not align exactly with those in the primary image. For example, marginally farther points on a column or nearer points on the back wall might become visible in the secondary scan. To account for this we modify Algorithm 1 slightly to pad the cell volume. The padding reduces the inner radial limit and increases the outer radial limit by as much as 0.5 m (or half the distance to the next excluded point, whichever is less).

From a computational point of view, our gridding process decomposes the scene into a two-dimensional grid (with one voxel in each azimuth and elevation direction); working with this grid is much more efficient than working with a conventional three-dimensional Cartesian grid. This computational benefit can speed up processing or allow for a



(a) Rendered scene (b) Wedge-shaped bins (c) Truncated voxels



(d) Radial distribution of points wedges in Fig. 3b

Fig. 3: Simulated four-lane highway

higher grid resolution for the same total processing time. Another advantage is that the number of points in each wedge-shaped bin (before shadow mitigation) is equal by design. This avoids a well-known limitation of Cartesian grids, where point density is much higher in near bins than far bins. These benefits of spherical gridding also apply as compared to cylindrical coordinates, as used in [24]. Cylindrical coordinates, like Cartesian coordinates, do not align with Lidar beams, making shadow mitigation difficult. Also, using cylindrical coordinates creates a computationally less efficient three-dimensional grid, where point densities are nonuniform across voxels.

V. Performance Evaluation

In this section we evaluate the degree to which our shadow-mitigation methods enhance algorithm performance. To this end, we performed computations using three algorithm variations: conventional ICET with a Cartesian grid, conventional ICET with shadow mitigation via ground-plane removal, and ICET with shadow mitigation via Algorithm 1. We compared performance using two simulated scenes. One scene was the roadway scene from Fig. 3, featuring sound-absorption walls beside the road and a series of 10 cylindrical pillars, two of which are shown in Fig. 3a. The second scene was an off-road terrain, shown in Fig. 4. For each scene, multiple Monte-Carlo trials were computed with randomized Lidar errors. Since true motion is known in the simulated environment, scan-matching estimates were compared to the truth to compute an error, which was characterized statistically over the set of Monte-Carlo trials. In all, our analysis considered 120 trials on each terrain (3 samples for scan pairs in each of 40 locations along the roadway, and 6 samples for scan pairs in each of 20 locations on the off-road terrain).

Scenes were created as solid bodies in AutoDesk Inventor and, with `extendedObjectMesh` in Matlab, converted to surface triangulations. Surfaces were sampled in Matlab using `monostaticLidarSensor` to form simulated Lidar point clouds. In an attempt to maintain consistency with other benchmarks, we calibrated our virtual Lidar sensor to match the specifications of the *Velodyne HDL-64E* used in the KITTI dataset [25]. For the roadway scene, the Lidar translates 20 m along the center of its lane at a constant velocity of $5\frac{m}{s}$. For the off-road scene, the Lidar translates at a constant forward velocity of $5\frac{m}{s}$ and and rotates at $30\frac{deg}{s}$ about the vertical axis, moving for 2.1 seconds and producing 20 pairs of frames sampled at 10Hz. Scan lines were generated assuming the Lidar is on a gimball (such that it does not tilt relative to the gravity vector). In the roadway scene, the columns along the roadway cause shadowing on the ground plane and vertical walls behind; in the off-road scene, the natural contours of the hills create terrain shadows, which are notoriously difficult to mitigate [26].

For performance-evaluation purposes, our shadow-matching algorithm was integrated with the ICET scan matching algorithm. ICET, as described in [19], predicts the scan error from first principles, in the absence of systematic biases. When shadow mitigation is successful, the probabilistic accuracy prediction of ICET should match the statistical accuracy obtained from the Monte Carlo simulation. By comparison, we expect the two accuracy metrics to diverge if shadow matching is not successful. For ICET analyses on Cartesian grids, cubic voxels were

used, with 3m edge length (a dimension tuned for favorable performance, so as not to artificially penalize the Cartesian analysis). For the roadway scene, the voxel grid was aligned with the vertical wall passing through the middle of associated voxels (noting that Cartesian methods perform better in this configuration than when walls clip through voxel corners).

VI. Results

After running Monte Carlo simulations, motion was estimated with scan matching. Errors were computed in each of six directions, including the translational directions $\{x, y, z\}$ and the rotational angles $\{\phi, \theta, \psi\}$, or roll, pitch and yaw. In each case the mean of the errors was computed and found to be negligibly small. Standard deviations for each error are tabulated below for the roadway scene, in Tab. I for the Cartesian grid analysis, in Tab. II for the Cartesian grid analysis with ground-plane removal, and in Tab. III for our proposed shadow-removal approach using spherical coordinates. Standard deviations are also tabulated for the offroad terrain scene, in Tab. IV for the Cartesian grid analysis and in Tab. V for our proposed shadow-removal approach. Ground-plane removal is ill-defined for the offroad scene, so results are not tabulated for that case.

In nearly all trials, the ICET algorithm converged. The exceptions occurred when the Cartesian grid was used for the roadway scene (with or without ground-plane removal). Convergence failures occurred either when the algorithm diverged entirely or when it failed to escape the local minima of the initialization. These situations were easily detectable, and they were omitted from our analysis. By comparison, computations on the spherical coordinate system generated by our new algorithm successfully converged in all trials on both scenes.

For the roadway scene, the effect of shadowing is most pronounced in the along-track (or x direction). Regardless of shadowing, translational errors are larger in the x-direction than in other directions (by a factor of about 5 times), because there is a relative dearth of features to mark progress along the road. For the Cartesian grids (with and without ground-plane removal), shadowing increases the actual x-direction errors substantially above the prediction (actual σ more than twice predicted value). By comparison, our new shadow-mitigation algorithm achieves an actual error that is very close to the predicted error in the x-direction (actual σ within 2% of the predicted value). Shadow mitigation with our spherical-grid approach is much more effective than with ground-plane removal in this case, because ground-plane removal addresses shadows that project on the ground but not those that project on the vertical walls. Not only does ground-plane removal fail to resolve vertical shadows, it also greatly increases vertical, pitch, and roll (z, ϕ, θ) errors, a result consistent with prior research, such as in [27, 28], which suggested re-introducing the ground-plane specifically to compute these states.

Similar trends are observed for the x-errors in the off-road terrain, with the Cartesian-grid actual errors much larger than predicted errors (by a factor of four) and with the spherical-grid actual errors close to predicted errors (within 13%). For the offroad scene, the actual errors in the other directions $(y, z, \phi, \theta, \psi)$ were also larger for the Cartesian grid, by a factor of 2-3 times, as compared to prediction;

TABLE I: Roadway - Cartesian grid (with ground plane)

	std error x (cm)	std error y (cm)	std error z (cm)	std error ϕ (deg)	std error (deg)	std error ψ (deg)
Actual*	0.978	0.0226	0.0197	0.00259	0.000897	0.00106
Predicted	0.467	0.0220	0.0175	0.00210	0.00125	0.00102

*27 of 120 frames rejected

TABLE II: Roadway - Cartesian grid (no ground plane)

	std error x (cm)	std error y (cm)	std error z (cm)	std error ϕ (deg)	std error (deg)	std error ψ (deg)
Actual**	1.151	0.025	0.124	0.0057	0.0126	0.00138
Predicted	0.437	0.0275	0.526	0.0236	0.0279	0.00113

**4 of 120 frames rejected

TABLE III: Roadway - Algorithm 1

	std error x (cm)	std error y (cm)	std error z (cm)	std error ϕ (deg)	std error (deg)	std error ψ (deg)
Actual	0.0958	0.0187	0.0149	0.00165	0.00159	0.000939
Predicted	0.0975	0.0189	0.0131	0.00172	0.00164	0.00108

TABLE IV: Offroad terrain - Cartesian grid

	std error x (cm)	std error y (cm)	std error z (cm)	std error ϕ (deg)	std error (deg)	std error ψ (deg)
Actual	0.644	0.253	0.180	0.00141	0.00735	0.00435
Predicted	0.166	0.117	0.0491	0.000932	0.00209	0.00236

TABLE V: Offroad terrain - Algorithm 1

	std error x (cm)	std error y (cm)	std error z (cm)	std error ϕ (deg)	std error (deg)	std error ψ (deg)
Actual	0.101	0.093	0.033	0.00079	0.00136	0.00162
Predicted	0.114	0.091	0.037	0.00074	0.00153	0.00186

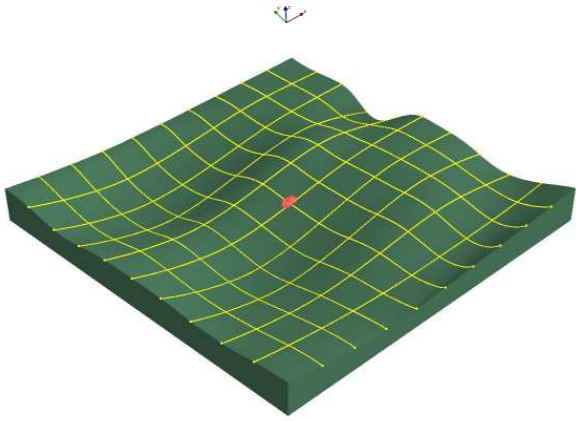


Fig. 4: Hilly off-road terrain (100x100m)

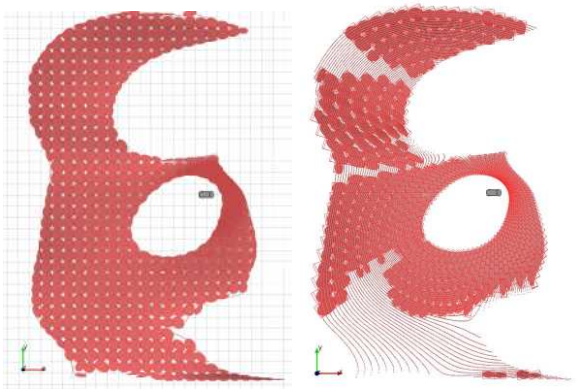


Fig. 5: Overhead view of Cartesian (left) and spherical (right) grids for offroad scene, with distribution ellipsoids (red) shown for each voxel used in the scan-matching computation

our new algorithm again predicted the actual errors well in these directions (within 15%).

For both scenes, perhaps surprisingly, the spherical grid resulted in higher accuracy than the Cartesian grid. We hypothesize this results from better conditioning due to more consistent clustering of points in each voxel, noting that the number of Lidar points varies little across the voxels on the spherical grid but varies substantially across the voxels in the Cartesian grid, as seen in Fig. 5.

VII. Discussion

As demonstrated by Tables I-V, our proposed shadow-mitigation strategy improves scan-matching accuracy by using a spherical voxel grid instead of a Cartesian one. The spherical voxels also greatly enhance the quality of accuracy predictions made by ICET in the presence of significant shadowing. As discussed above, other benefits of the spherical grid include reduced computational effort (due to the use of a 2D grid in place of a 3D grid) and less sensitivity to grid placement. Grid sensitivity can be visualized at the edges of the left image in Fig. 5, where ellipsoids are poorly defined because relative few scan points appear in voxels far from the Lidar.

Despite the many advantages of the spherical grid introduced as part of Algorithm 1, there are likely to be several limitations, which we will explore in future work. We suspect one disadvantage of our spherical-grid approach is that existing search-acceleration strategies [29] will work poorly for the case of large initial uncertainty, such as in the lost robot problem where initial conditions are not known. Another limitation is that our method only addresses errors caused by shadowing. The proposed algorithm does not address other error sources (such as moving objects and perspective shifts), and these anomalies would be expected to cause differences between true and predicted σ-values for error in many scenes.

VIII. Conclusion

This paper introduced a new method for mitigating shadowing errors in Lidar scan matching. This shadow-matching algorithm runs as a preprocessing step before scan matching by identifying the nearest meaningful cluster of points in a set of azimuth and elevation bins, in effect creating a 2D voxel grid in spherical coordinates. This grid aligns with Lidar rays, and with shadow edges; thus, shadow edges can be eliminated by preserving only the nearest point cluster in each wedge-shaped azimuth/elevation bin. As compared to the commonly used technique of ground-plane removal, our proposed method offers advantages in that it works even on oblique or uneven terrain and in that it retains useful information needed to compute vertical, pitch, and roll states accurately. For voxel-based scan matching methods like NDT or ICET, the resulting grid can be reused by the scan matching algorithm. The 2D spherical grid offers several advantages over a 3D Cartesian grid for scan matching including reduced computational costs and better numerical conditioning due to more uniform balancing of the number of points in each voxel. These conclusions were verified through Monte Carlo simulations of urban and offroad terrains.

IX. Acknowledgements

The authors wish to acknowledge and thank the U.S. Department of Transportation Joint Program Office (ITS JPO) and the Office of the Assistant Secretary for Research and Technology (OST-R) for sponsorship of this work. We also gratefully acknowledge NSF grant CNS-1836942, which supported specific aspects of this research. Opinions discussed here are those of the authors and do not necessarily represent those of the DOT, NSF, or other affiliated agencies.

References

[1] Q. Li et al. “LO-Net: Deep Real-Time Lidar Odometry”. In: *2019 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*. June 2019, pp. 8465–8474. DOI: 10.1109/CVPR.2019.00867.

[2] Younggun Cho, Giseop Kim, and Ayoung Kim. “Unsupervised Geometry-Aware Deep LiDAR Odometry”. In: *2020 IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2020, pp. 2145–2152. DOI: 10.1109/ICRA40945.2020.9197366.

[3] Oliver Willers et al. “Safety Concerns and Mitigation Approaches Regarding the Use of Deep Learning in Safety-Critical Perception Tasks”. In: *Computer Safety, Reliability, and Security. SAFECOMP 2020 Workshops*. Ed. by António Casimiro et al. Cham: Springer Int. Publishing, 2020, pp. 336–350. ISBN: 978-3-030-55583-2.

[4] Zhiyuan Zhang, Yuchao Dai, and Jiadai Sun. “Deep learning based point cloud registration: an overview”. In: *Virtual Reality & Intelligent Hardware* 2.3 (2020). 3D Visual Processing and Reconstruction Special Issue, pp. 222–246. ISSN: 2096-5796. DOI: 10.1016/j.vrih.2020.05.002.

[5] P. J. Besl and N. D. McKay. “A method for registration of 3-D shapes”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256. DOI: 10.1109/34.121791.

[6] Jan Elseberg, Dorit Borrmann, and Andreas Nüchter. "One billion points in the cloud – an octree for efficient processing of 3D laser scans". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 76 (2013). Terrestrial 3D modelling, pp. 76–88. ISSN: 0924-2716. DOI: 10.1016/j.isprsjprs.2012.10.004.

[7] Albert Palomer et al. "Bathymetry-based SLAM with difference of normals point-cloud subsampling and probabilistic ICP registration". In: *2013 MTS/IEEE OCEANS - Bergen*. 2013, pp. 1–8. DOI: 10.1109/OCEANS-Bergen.2013.6608091.

[8] Oren Dovrat, Itai Lang, and Shai Avidan. "Learning to Sample". In: *Proc. of the IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[9] Haibo Sun et al. "Vision Odometer Based on RGB-D Camera". In: *2018 Int. Conf. Robots & Intelligent System (ICRIS)*. 2018, pp. 168–171. DOI: 10.1109/ICRIS.2018.00052.

[10] Joe Khalife, Sonya Ragothaman, and Zaher M Kassas. "Pose estimation with lidar odometry and cellular pseudoranges". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 1722–1727.

[11] Ji Zhang and Sanjiv Singh. "LOAM: Lidar Odometry and Mapping in Real-time." In: *Robotics: Science and Systems*. Vol. 2. 9. 2014.

[12] Tixiao Shan and Brendan Englot. "LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain". In: *2018 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*. Oct. 2018, pp. 4758–4765. DOI: 10.1109/IROS.2018.8594299.

[13] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. "Generalized-icp." In: *Robotics: science and systems*. Vol. 2. 4. Seattle, WA. 2009, p. 435.

[14] Y. S. Park, H. Jang, and A. Kim. "I-LOAM: Intensity Enhanced LiDAR Odometry and Mapping". In: *2020 17th Int. Conf. Ubiquitous Robots (UR)*. 2020, pp. 455–458. DOI: 10.1109/UR49135.2020.9144987.

[15] Zhichao Li and Naiyan Wang. "DMLO: Deep Matching LiDAR Odometry". In: *2020 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*. 2020, pp. 6010–6017. DOI: 10.1109/IROS45743.2020.9341206.

[16] P. Biber and W. Straßer. "The Normal Distributions Transform: A New Approach to Laser Scan Matching". In: *IEEE Int. Conf. Intelligent Robots and Systems*. Vol. 3. Nov. 2003, 2743–2748 vol.3. DOI: 10.1109/IROS.2003.1249285.

[17] Su Pang et al. "3D Scan Registration Based Localization for Autonomous Vehicles - A Comparison of NDT and ICP under Realistic Conditions". In: *2018 IEEE 88th Vehicular Technology Conf. (VTC-Fall)*. 2018, pp. 1–5. DOI: 10.1109/VTCFall.2018.8690819.

[18] M. McDermott and J.H. Rife. "Enhanced Laser-Scan Matching with Online Error Estimation for Highway and Tunnel Driving". In: *Proc. 2022 Int. Technical Meeting of the Institute of Navigation*. Jan. 2022, pp. 643–654. DOI: 10.33012/2022.18249.

[19] M. McDermott and J.H. Rife. "Validation of ICET Performance Characterization for Geometric-based Laser-scan Matching of 3D Point Clouds". In: *in press*. 2022.

[20] Shinsuke Shimojo, Gerald H Silverman, and Ken Nakayama. "Occlusion and the solution to the aperture problem for motion". In: *Vision research* 29.5 (1989), pp. 619–626.

[21] Angel X Chang et al. "Shapenet: An information-rich 3d model repository". In: *arXiv preprint arXiv:1512.03012* (2015).

[22] Bichen Wu et al. "SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud". In: *2018 IEEE Int. Conf. Robotics and Automation (ICRA)*. 2018, pp. 1887–1893. DOI: 10.1109/ICRA.2018.8462926.

[23] D.F. Huber and M. Hebert. "A new approach to 3-D terrain mapping". In: *Proceedings 1999 IEEE/RSJ Int. Conf. Intelligent Robots and Systems.* Vol. 2. 1999, 1121–1127 vol.2. DOI: 10.1109/IROS.1999.812830.

[24] Martijn Heller, Nikita Petrov, and Alexander Yarovoy. "A Radar-Oriented Approach to the Normal Distributions Transform". In: *2021 18th European Radar Conf. (EuRAD)*. 2022, pp. 165–168. DOI: 10.23919/EuRAD50154.2022.9784492.

[25] Andreas Geiger et al. "Vision meets Robotics: The KITTI Dataset". In: *Int. J. Robotics Research (IJRR)* (2013).

[26] Håkan Almqvist et al. "Improving point-cloud accuracy from a moving platform in field operations". In: *2013 IEEE Int. Conf. Robotics and Automation (ICRA)*. 2013, pp. 733–738. DOI: 10.1109/ICRA.2013.6630654.

[27] Hyungjin Kim, Seungwon Song, and Hyun Myung. "GP-ICP: Ground plane ICP for mobile robots". In: *IEEE Access* 7 (2019), pp. 76599–76610.

[28] Gaurav Pandey, Shashank Giri, and Jame R. Mcbride. "Alignment of 3D point clouds with a dominant ground plane". In: *2017 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*. 2017, pp. 2143–2150. DOI: 10.1109/IROS.2017.8206031.

[29] Erik Einhorn and Horst-Michael Gross. "Generic NDT mapping in dynamic environments and its application for lifelong SLAM". In: *Robotics and Autonomous Systems* (2015). DOI: 10.1016/j.robot.2014.08.008.