LiDAR Reflection Recovery via Shadow Boxing

Kyungmin Kim
Department of Electrical and Electronics Engineering,
Chung-Ang University,
Seoul, Republic of Korea
kyungddin@cau.ac.kr

Jeongyeup Paek
Department of Computer Science and Engineering,
Chung-Ang University,
Seoul, Republic of Korea
jpaek@cau.ac.kr

***Abstract*—*Light detection and ranging* (LiDAR) sensor is a device that can effectively collect 3D spatial information using lasers. However, reflective surfaces like mirrors pose a challenge for LiDAR, making it difficult to distinguish between objects in front and behind a mirror and their reflected counterparts. In this paper, we propose the *Shadow Boxing Method*, a technique for effectively classifying reflected points that appear to be behind the mirror. We will differentiate reflected points from points behind the mirror, and further restore these identified points to their original 3D locations by accurately detecting the relative position of the mirror and the angle of reflections. This will overcome the existing limitations of LiDAR sensors, enabling more accurate sensing. We envision that our idea can potentially improve the safety of LiDAR-based autonomous vehicles even when confronted with highly reflective surfaces such as traffic safety mirrors.**

***Index Terms*—LiDAR, Mirror Reflection, Point Cloud, 3D Sensing, Object Detection, Segmentation**

I. Introduction

Light detection and ranging (LiDAR) sensor is a device that acquires spatial information of surrounding objects through laser pulses. LiDAR is actively used in fields like robotics and autonomous driving (e.g. including robot vacuum cleaners) due to its ability to effectively obtain 3D object and environment information compared to other sensors [1]–[4]. However, LiDAR can be considered vulnerable in situations where lasers are reflected or partly penetrated, such as with mirrors or glasses. In the case of mirrors, most of the laser pulses are reflected, and if the reflected laser hits another object, it creates a problem by generating point clouds at incorrect locations. This can cause hallucination in object recognition or SLAM mapping [5].

A few prior works have investigated this challenge. For example, Li et al. [6] improve SLAM accuracy by classifying reflective surfaces through a geometric approach. However, this method focuses on offline map refinement, showing limitations in real-time point restoration. Furthermore, Zhao and Schwertfeger [7] presented an open 3D dataset and benchmark for deep learning-based research on detecting and classifying reflective surfaces. Yet, research on restoring and utilizing distorted points in real-time remains in its early stages.

"This research was supported by the MSIT(Ministry of Science, ICT), Korea, under the National Program for Excellence in SW), supervised by the IITP(Institute of Information & communications Technology Planning & Evaluation) in 2025"(2025-0-00032). J. Paek is the corresponding author.

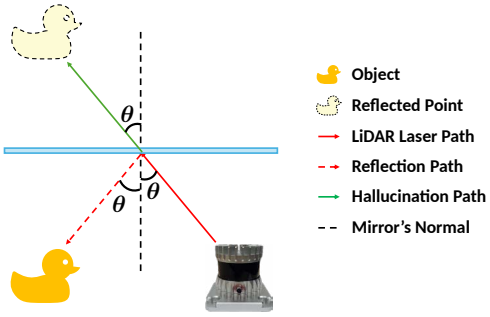


Fig. 1: Reflection Hallucination

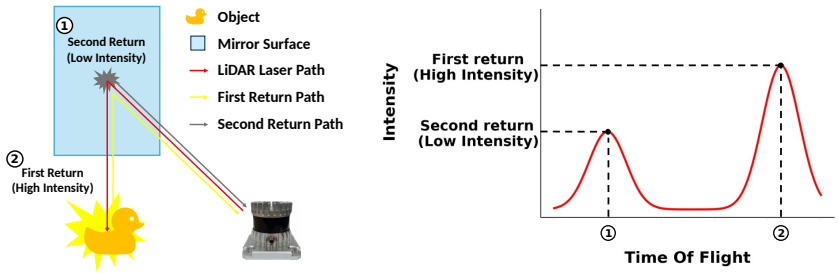
If the reflected 3D point clouds are utilized well, it is possible to obtain information about the sides or back of an object that the LiDAR sensor cannot originally detect. In other words, by understanding the reflection, it is possible not only to correct LiDAR's erroneous information but also to expand the sensor's perception range and direction. To this end, this paper proposes the *Shadow Boxing Method*, a technique for effectively detecting mirrors and restoring reflected points beyond them. We evaluate our proposal via real-time experiments using Ouster's OS1-32 Rev6 LiDAR equipment [8].

II. Background

In this section, we describe the basics of mirror reflection and dual return, which are two key concepts for understanding our shadow box generation method.

A. Mirror Reflection

Flat mirrors induce specular reflection, where the reflection angle of a light ray equals its incidence angle. When a LiDAR beam strikes a mirror, its path is altered, but the sensor fails to recognize this deflection, perceiving the laser as having traveled straight through the mirror's surface. This phenomenon generates an erroneous *hallucinated* point cloud behind the mirror, as illustrated in Fig. 1. The trajectory of this reflected point cloud is governed by precise geometric rules: the horizontal (azimuthal) angle is preserved, while the vertical (altitudinal) angle depends on the mirror's inclination. We leverage this predictable characteristic to define a *Shadow Box* for classifying these reflected points.



(a) Dual Return Path

(b) Dual Return Path Graph

Fig. 2: Illustration of the LiDAR dual return mechanism on a mirror. (a) The path of a single laser beam generating two returns. (b) The resulting graph of intensity versus time-of-flight for each return.

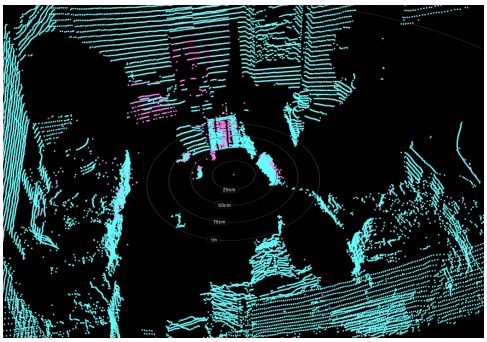


Fig. 3: Second Return Visualization

B. Dual Return

Some LiDAR sensors support *dual return mode*. Dual return is a feature that distinguishes multiple return values when a single laser hits several objects due to reflection or penetration. This mode can contribute effectively to mirror detection. When a laser hits a mirror, most of the light is reflected to hit another object, with only a slight loss of energy. This process, illustrated in Fig. 2(a), results in the object forming a strong signal (the first return), while the mirror surface itself creates a weak signal (the second return).

The method of distinguishing between the first and second return varies by hardware manufacturer. In the case of Ouster [9], multiple signals generated from a single laser are classified into return types based on their intensity. This relationship is visualized in Fig. 2(b), where the first return exhibits a higher intensity despite a longer time-of-flight, while the second return from the closer mirror surface shows a lower intensity.

Algorithm 1 Mirror Plane Detection

Require: Point clouds P_1 (first return), P_2 (second return)
Ensure: Mirror pose $\mathcal{M}$ (center, normal, extent) or null

```
Clusters ← DBSCAN(P_2, eps, min_pts)
for each cluster C in Clusters do
    if len(C) < points_threshold then
        continue
    end if
    planarity ← CalculatePlanarityPCA(C)
    if planarity < planarity_threshold then
        PlaneModel, Inliers ← RANSAC(C, dist_thresh)
        if len(Inliers) > inlier_threshold then
            PlaneModel ← RefinePlaneWithNeighbors(P_1, Inliers)
            M ← ComputePoseFromPlane(PlaneModel)
            return M
        end if
    end if
end for
return null
```

III. Design and Methodology

This section describes the design of the step-by-step method for effective mirror detection and reflected point restoration.

A. Mirror Detection

LiDAR's dual return mode is actively used for accurate mirror detection. Although Ouster's OS1-32 officially supports dual return, second returns account for a very small proportion, only 0.4%, of the total points as shown in Fig. 3. In the visualization, first returns are denoted by blue points, whereas second returns are indicated by pink points.

However, as described in §II-B, second returns exhibit a distinct characteristic on mirror surfaces: a dense concentration at the center. This key feature distinguishes them from the sparse clusters or noise found in other cases. Thus, even though these points are few, their density allows us to effectively isolate them for mirror detection. To implement this, we employ the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm [10]. This method effectively groups the densely concentrated second return points, while simultaneously filtering out sparse noise and clusters with an insufficient number of points.

Next, we search for a plane at the location of these refined second return points. We assume the mirror is rectangular and has a frame. In this case, a plane will exist where the first return forms the boundary and the second return forms the surface. Therefore, we use PCA to calculate the planarity of the point cloud at the second point locations to further reduce candidates, and then use the RANSAC Algorithm to finally find and classify the mirror [11], [12].

Once the mirror is classified within the point cloud, the next step is to determine the front and back of the mirror. Since the front of the mirror is closer to the sensor, the direction facing the mirror on the plane is set as the front, and the opposite direction is set as the back. The back-side information thus obtained contributes to the creation of the *shadow box*.

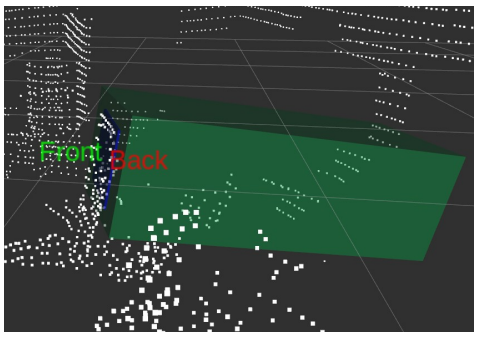


Fig. 4: Shadow Boxing Result

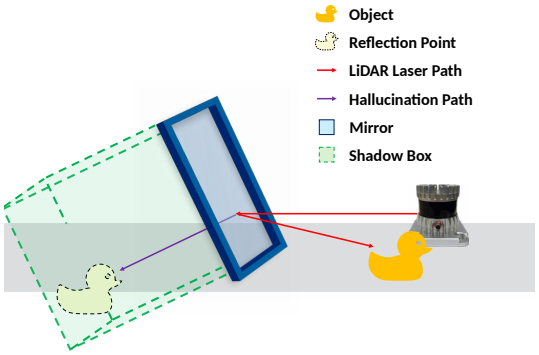


Fig. 5: Shadow Box's Direction

B. Finding Reflection Points

After obtaining the back-side directional information, the next task is to distinguish between reflected points and ordinary points in the point cloud that exists behind the mirror. §II-A established that no object's point cloud can normally be generated if we follow the laser's path into the space behind the mirror.

Applying this concept, we create a bounding box in the direction of the laser's path behind the mirror and consider the point cloud inside this box to be reflected points. The width and height of the bounding box are determined based on the size of the mirror surface, as illustrated in Fig. 4. However, there are cases where the bounding box cannot cover all points. In such cases, we are able to restore points outside the box using a Kd-Tree [13].

The next important point is the *tilt* of the mirror. If the mirror is tilted, the reflected points are formed along that tilt, causing a distortion in the z-axis position. This concept is visually detailed in Fig. 5 However, the shadow boxing method has the advantage of being able to effectively detect this distortion because it forms a bounding box in the space behind the mirror relative to the mirror's surface.

C. Filtering and Mirroring Back Point Cloud

After classification, these reflected points are restored to their original positions, and the existing erroneous points are filtered out. The Householder Transformation [14] is used to restore them to their original locations. However, using only the Householder Transformation cannot effectively restore the distortion caused by the mirror's tilt. Therefore, for the final position, tilt interpolation is performed by inverting the z-axis information of the mirror's tilt.

Algorithm 2 Reflected Point Identification

Require: Mirror pose $\mathcal{M}$, Point cloud P_1
Ensure: Reflected point cloud $P_{\text{reflected}}$
1: ShadowBox $\leftarrow$ ConstructSearchVolume($\mathcal{M}$)
2: $I_{\text{initial}} \leftarrow$ FindPointsInVolume(P_1, ShadowBox)
3: kdtree $\leftarrow$ BuildKdTree(P_1)
4: $I_{\text{neighbors}} \leftarrow$ RadiusSearch(P_1, kdtree, I_{initial}, r)
5: $I_{\text{candidate}} \leftarrow I_{\text{initial}} \cup I_{\text{neighbors}}$
6: $P_{\text{candidate}} \leftarrow P_1[I_{\text{candidate}}]$
7: $P_{\text{reflected}} \leftarrow$ CullPointsNearMirror($P_{\text{candidate}}, \mathcal{M}, d_{\text{cull}}$)
8: **return** $P_{\text{reflected}}$

Formally, let $\mathbf{P}_{orig}$ denote the original point to be restored, $\mathbf{P}_{restored}$ the final restored point, $\mathbf{n} = [n_x, n_y, n_z]^\top$ the unit normal vector of the mirror plane (pointing away from the mirrored surface), and $\mathbf{Q}$ a point on the mirror plane (e.g., the mirror center). Let I be the 3×3 identity matrix and k_z the z-axis correction factor.

Step 1: Householder Reflection The Householder reflection matrix is defined as

$$H = I - 2\mathbf{n}\mathbf{n}^\top \tag{1}$$

and the reflected position of $\mathbf{P}_{orig}$ with respect to the mirror plane is

$$\mathbf{P}_{reflected} = H(\mathbf{P}_{orig} - \mathbf{Q}) + \mathbf{Q}. \tag{2}$$

Step 2: Z-axis Tilt Correction To compensate for the distortion caused by the mirror's tilt, a correction vector $\mathbf{c}$ along the z-axis is applied:

$$\Delta z = -n_z \cdot k_z, \qquad \mathbf{c} = \begin{bmatrix} 0 \\ 0 \\ \Delta z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -n_z \cdot k_z \end{bmatrix}. \tag{3}$$

Final Restored Position The final restored point is then computed as,

$$\begin{aligned} \mathbf{P}_{restored} &= \mathbf{P}_{reflected} + \mathbf{c} \\ &= (I - 2\mathbf{n}\mathbf{n}^\top)(\mathbf{P}_{orig} - \mathbf{Q}) \\ &\quad + \mathbf{Q} + \mathbf{c}. \end{aligned} \tag{4}$$

This formulation ensures that the points reflected by the mirror are restored to their original 3D positions while simultaneously correcting distortions due to the mirror's tilt.

D. GPU Acceleration

To meet the real-time processing requirements essential for LiDAR applications, we apply GPU acceleration to the primary computational bottlenecks of our algorithm. The implementation leverages both Open3D's tensor-based and PyTorch[1] to execute parallel operations on the NVIDIA CUDA[2]

[1]PyTorch, https://pytorch.org/

[2]CUDA (Compute Unified Device Architecture), https://developer.nvidia.com/cuda-zone

(a) Front View **(b)** Back View **(c)** Blocking Experiment

Fig. 6: Environment Setting

(a) Air Purifier **(b)** Duck Doll **(c)** Person

Fig. 7: Experiment Target Objects

platform. The key acceleration points reflected in our current implementation are as follows:

- **DBSCAN Clustering:** The second return point cloud is first converted to an Open3D GPU tensor object. This allows the DBSCAN clustering to be performed directly on the GPU, enabling the rapid identification of dense point regions corresponding to the mirror surface.
- **RANSAC Plane Segmentation:** For the clusters identified by DBSCAN, the search for the mirror plane is also accelerated using Open3D's GPU-based RANSAC function.
- **Householder Transformation:** In the final point restoration stage, the most computationally expensive operation—the Householder transformation—is implemented using PyTorch tensors. The points targeted for restoration are transferred to the GPU, where the matrix and vector calculations are processed in parallel before the results are returned to the CPU.

By offloading these core bottlenecks—clustering, plane segmentation, and matrix transformations—to the GPU, we significantly improve the overall throughput of the pipeline, laying the groundwork for real-time operation.

IV. Evaluation

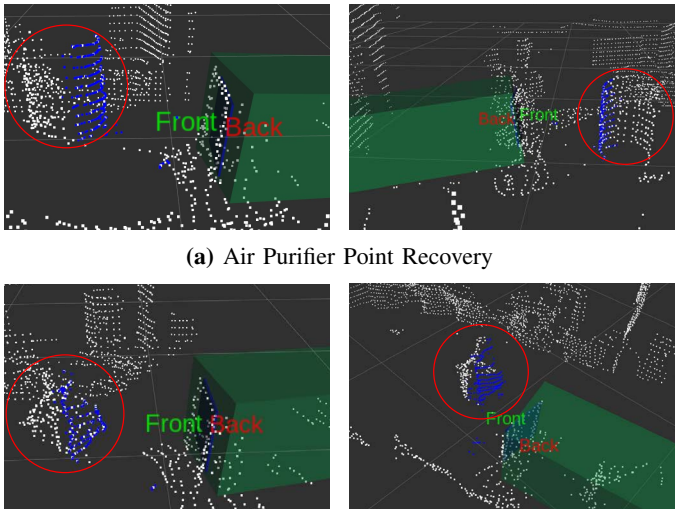
The equipment used for the experiment is a single Ouster OS1-32 Rev6. Additionally, Robot Operating System (ROS)[3] and the Open3D[4] library are used in the process of handling the point cloud. ROS automatically parses the return values obtained by the LiDAR, and Open3D allows effective point cloud processing.

Two experiments are conducted in a typical indoor laboratory environment, where a mirror is placed near the LiDAR sensor to cause specular reflection as shown in Fig. 6. To verify that the restoration of reflections is effective, we perform restoration and detection experiments on various objects including an air purifier, a duck doll, and a person (Fig. 7).

The first experiment focuses on point cloud recovery. Under normal conditions, the LiDAR can only detect one side of the three test objects. By utilizing the mirror's reflection, we are able to reconstruct the point cloud of the opposite, non-visible surfaces. The results of this experiment are presented in Fig. 8.

The second experiment investigates detection under conditions of occlusion. We examine whether the far side of an object could be perceived via reflection even when the LiDAR's direct line of sight is obstructed. As shown in Fig. 9, the sensor is able to effectively detect the occluded person surface through reflection, confirming the method's viability even with a partially blocked sensor view.

Finally, to validate the real-time performance of our method, we measure the average processing speed in Frames Per



(a) Air Purifier Point Recovery

(b) Duck Doll Point Recovery

Fig. 8: Recovery experiment results for three objects. White points represent first returns, blue points indicate recovered points, and the red circle highlights the final shape of the recovered object.

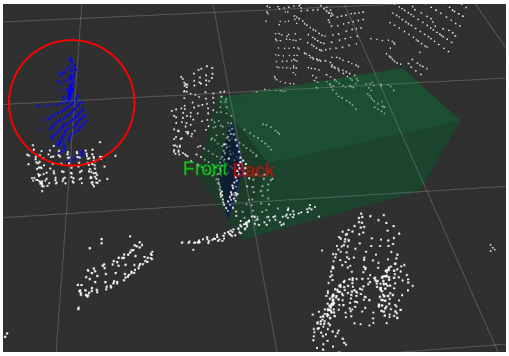


Fig. 9: Blocking experiment result. A red circle highlights the restored point cloud for the occluded side of the person.

[3]Robot Operating System (ROS) , https://www.ros.org/
[4]Open3D, http://www.open3d.org/

TABLE I: Performance Comparison (FPS)

Implementation	Scenario	Average FPS
CPU-only	Non-Detection	10.0
	Detection	**2.7**
GPU-accelerated	Non-Detection	10.0
	Detection	**8.8**

Second (FPS) under two distinct scenarios: 1) a baseline without a mirror present (Non-Detection) and 2) with a mirror being actively processed (Detection). The experiments are conducted on a system equipped with an Intel Core i7-10700F CPU and an NVIDIA GeForce RTX 2060 GPU. The results, summarized in Table I, highlight the impact of the computational load and the effectiveness of GPU acceleration. In the Non-Detection scenario, both implementations operate at approximately 10.0 FPS, matching the native frequency of the Ouster OS1-32 sensor. This indicates that the baseline processing overhead is minimal.

However, a significant performance difference emerges under the Detection workload. The CPU-based implementation's performance drops sharply to an average of 2.7 FPS, rendering it unsuitable for real-time applications. In contrast, our proposed GPU-accelerated version maintains a high frame rate of 8.8 FPS, successfully processing the data at a rate close to the sensor's output. This represents a performance increase of over 3.2 times under load and confirms that GPU acceleration is crucial for the real-time viability of our method.

The experimental results validate that our proposed shadow boxing method not only successfully utilizes reflections to expand the LiDAR's detection capabilities, but also achieves this in real-time thanks to GPU acceleration. This combination of functional robustness and high performance confirms the viability of our approach for practical applications.

V. Conclusion

In this paper, we restored mirror reflected points to reconstruct the rear side of objects in 3D point clouds by utilizing the reflection characteristics of LiDAR lasers. Through real experiments, it was possible not only to effectively distinguish reflected points from points behind the mirror, but also to correct for tilt distortion. However, some limitations exist. Since only a single 32-channel LiDAR unit was used, the quantity of second returns varied significantly depending on the angle of the mirror. This issue can be improved by using higher-end sensors or multiple sensors [15]. Furthermore, generalizability of our scheme should be further validated through more experiments with diverse environments, objects, mirror types, and LiDARs. As future work, we plan to apply our scheme to a real vehicular setting. We believe that by overcoming the limitations of LiDAR sensors with respect to reflective surfaces, our proposed algorithm will contribute to the fields of object recognition, SLAM mapping, and autonomous driving in the future.

References

[1] Y. Li and J. Ibanez-Guzman, "Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems," *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 50–61, 2020.

[2] J. Ryu and J. Paek, "Poster: Fast field-of-view expansion for collaborative object detection," in *The 22nd ACM International Conference on Mobile Systems, Applications, and Services (MobiSys'24)*. ACM, June 2024.

[3] C. S. Shin, W. Pang, C. Li, F. Bai, F. Ahmad, J. Paek, and R. Govindan, "Recap: 3d traffic reconstruction," in *Proceedings of The 30th Annual International Conference on Mobile Computing and Networking (MobiCom'24)*. ACM, Nov. 2024, pp. 1252–1267. [Online]. Available: https://dl.acm.org/doi/10.1145/3636534.3690691

[4] M. Choi, J. Ryu, Y. Son, S. Cho, and J. Paek, "LiDAR-based Localization for Autonomous Vehicles - Survey and Recent Trends," in *The 15th International Conference on Information and Communication Technology Convergence (ICTC) - Workshop on Big Data and 5G&6G Communication Networks (IWBCN)*. IEEE, Oct 2024, pp. 456–460.

[5] S. Fang and H. Li, "Multi-Vehicle Cooperative Simultaneous LiDAR SLAM and Object Tracking in Dynamic Environments," *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[6] Y. Li, X. Zhao, and S. Schwertfeger, "Detection and Utilization of Reflections in LiDAR Scans through Plane Optimization and Plane SLAM," *Sensors*, vol. 24, no. 15, 2024.

[7] X. Zhao and S. Schwertfeger, "3DRef: 3D Dataset and Benchmark for Reflection Detection in RGB and Lidar Data," 2024. [Online]. Available: https://arxiv.org/abs/2403.06538

[8] Ouster, Inc., "OS1 LiDAR Sensor," https://ouster.com/ko/products/hardware/os1-lidar-sensor, accessed: 2025-08-20.

[9] ——, "Ouster Sensor Docs," https://static.ouster.dev/sensor-docs/index.html#, 2025, accessed: 2025-08-20.

[10] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*. AAAI Press, 1996, pp. 226–231.

[11] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, vol. 2, no. 11, pp. 559–572, 1901.

[12] 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.

[13] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[14] A. S. Householder, "Unitary triangularization of a nonsymmetric matrix," *Journal of the ACM (JACM)*, vol. 5, no. 4, pp. 339–342, 1958.

[15] C. Henley, S. Somasundaram, J. Hollmann, and R. Raskar, "Detection and mapping of specular surfaces using multibounce LiDAR returns," *Optics Express*, vol. 31, no. 4, p. 6370, Feb. 2023.