IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 22, 2019, accepted March 16, 2019, date of publication April 10, 2019, date of current version April 16, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2909467

Reflective Fiducials for Localization With 3D Light Detection and Ranging Scanners

SPENCER DAVIS, KENNETH G. RICKS, AND RYAN A. TAYLOR
Electrical and Computer Engineering, The University of Alabama at Tuscaloosa, Tuscaloosa, AL 35487, USA
Corresponding author: Spencer Davis (dsdavis1@crimson.ua.edu)

ABSTRACT Fiducial markers are commonly used to localize robots. Most existing systems use standard cameras to detect simple patterns on flat tags. Since depth is not directly sensed, the pose must be inferred from the tag geometry. These systems are low-cost and easy to implement on robotic systems, but their performances suffer in low-light conditions and on computationally constrained processors. We propose using 3D light detection and ranging (LiDAR) scanners to mitigate these issues. The reflectivity measurement provided by most LiDAR sensors provides a simple way to discern geometric differences on surfaces. We utilize this fact to create a custom "beacon" with reflective fiducials. Next, we design a high-performance segmentation and localization algorithm to find the 2D pose of a mobile robot. Our experiments proved that our system achieves an average euclidean error of less than 0.063 m at ranges of over 10 m while maintaining a runtime of under 3 ms on a basic single board computer. Additionally, our system is highly occlusion resistant. These results are confirmed with multiple field tests of the system.

INDEX TERMS 3D LiDAR, computer vision, fiducial based localization, mobile robots, robotics and automation, robot sensing systems.

I. INTRODUCTION

Autonomous robots must be able to discern their positions and orientations within an environment. Cameras are often used in conjunction with a visual marker to provide this feedback. These systems rely on a variety of marker types, including visual fiducials [1], reflective fiducials [2], and actively lit markers [3]. These methods provide a low cost and unintrusive (little environmental modification required) way to localize a robot or other object within its environment. However, camera-based systems are subject to a common set of problems. They are dependent on ambient lighting conditions and generally (in the case of monocular systems) cannot directly sense depth. Since depth must be inferred, these algorithms are computationally complex and subject to image distortion.

Three-dimensional light detection and ranging (LiDAR) scanners could be applied to this problem to mitigate these issues. A fiducial localization system based on LiDAR could be used in the same ways as visual methods are: to provide ground truth data for testing, to enable complete localization for a system operating within line-of-site in a confined area, and/or to correct error induced from other methods after the robot has left and re-entered the range of the fiducial system. However, the use of LiDAR would allow operation of the system in all lighting conditions and eliminate the image processing necessary for depth inference. This is shown in [4], where a 2D Hokuyo LiDAR sensor is used to provide a homing reference to an autonomous sample return rover, and [5], where a custom 3D LiDAR scanning system is used to localize an unmanned aerial vehicle.

In this paper, we introduce a method of measuring 2D pose relative to a passive and stationary beacon using a 3D LiDAR sensor. We exploit the reflectivity measurement present in the data output of most 3D LiDAR scanners to segment a highly reflective beacon in real-time from sparse point cloud data. Then, we use the beacon geometry to calculate the 2D pose of the robot in the beacon frame of reference. The result is a versatile localization system that scales effectively to different range requirements, is highly invariant to lighting conditions, and can operate despite significant occlusion of the beacon (unlike other common fiducial systems [6]). Our beacon design is easy to manufacture and requires no active components. Additionally, our system is computationally efficient and capable of operating at the full scanning frequency of most LiDAR scanners while utilizing only a small fraction of available CPU time.

The associate editor coordinating the review of this manuscript and approving it for publication was Yu-Huei Cheng.

II. RELATED WORK

Robotic localization systems that use inertial and wheel sensors can provide accurate poses over a short period of time. However, since these methods rely on the integration of error-prone measurements, their pose estimates will drift from the true value [7]. Additionally, these methods are dependent on the starting pose of the robotic system and do not account for the kidnapped robot problem [8]. Global positioning systems can provide solutions to these problems, but some systems operate in GPS-denied environments. For these reasons, systems which provide localizations relative to one or more fixed markers within the environment have been researched extensively.

Early work in this domain focused on deriving the position of robots using time-of-flight or orientation estimates between a robot and multiple beacon nodes. Ultrasonics are used in [9] to measure the time-of-flight between a set of stationary transducers and robot with several receivers. In [10], Kurth et al. use radio beacons to determine the position of a robot with time-of-flight measurements. A rotational optical sensor is employed to detect stationary infrared landmarks in [11].

As the cost of cameras and computational power decreased, research began to focus on visual methods. These methods are typically more flexible than the older range and heading based systems are because they can extract more information from the artificial markers. In some cases, a full six degrees of freedom localization can be derived with a single marker. These systems can be categorized into those which use active markers such as LEDs, including [2], [3], [12], [13] and those which are completely passive [1], [14]–[19]. In [1], visual tags which contain coded information are localized with full six degrees of freedom in the environment. In this work, Olson et al. focus on improving the localization accuracy and coding system of other fiducial systems (such as [19]). In [14], stereo vision is used to triangulate visual markers. The markers in this work are circular and have colored rectangular borders capable of encoding information. A camera with a fisheye lens is used to localize a robot based on two artificial landmarks in [15]. The landmarks are cylinders (instead of the more common flat tags) and the system is capable of deriving both range and heading from each landmark. In [17], ceiling mounted markers are used in conjunction with a robot-mounted pan-and-tilt camera mechanism. In this work, lighting differences are identified as a key issue in the identification of markers and an iterative algorithm is introduced to mitigate this problem. Visual fiducials consisting of concentric circles have been proposed [16]. In [18], ceiling mounted alphanumeric character-shaped landmarks are used to localize a robot indoors.

Passive methods suffer from degraded performance in low-light conditions. For this reason, many systems use active lighting. In [12], Buchan et al. use a monocular camera to detect LEDs and localize swarms of underwater vehicles by solving a three-point PnP problem. A dynamic vision sensor is used to localize an unmanned aerial vehicle (UAV) by detecting LEDs that blink at a high frequency in [13]. A monocular camera and several LEDs are used to provide a six degrees of freedom localization of an UAV in [3]. In this system, variable numbers of LEDs can be mounted on the UAV to increase the system robustness. In [2], triangular, retro-reflective markers are used in conjunction with a monocular camera to localize a robot in two dimensions. The camera is surrounded with infrared LEDs to illuminate the reflective marker.

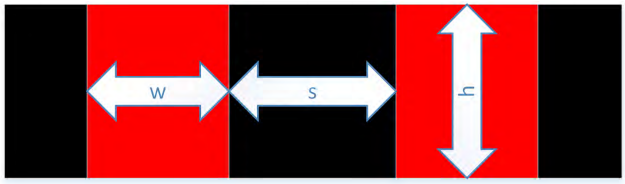

FIGURE 1. Passive beacon design (red is retro-reflective, black is matte).

Camera systems have been more thoroughly researched than LiDAR based systems for this application. LiDAR based systems are presented in [4] and [5]. In [5], an UAV is identified using a 2D LiDAR sensor. The scanner is constantly moving to provide low-frequency 3D scans. This work uses the reflectivity measurement from the scanner to determine the UAV orientation. In [4], a 2D scanner is used to provide a homing reference by detecting three equally-spaced vertical bars. The direct depth measurement provided by the scanner allows easy calculation of the robot position. However, the intensity of the LiDAR returns is not used to facilitate the marker identification. Instead, the gaps between the vertical bars are used to ensure accurate identification. This makes the marker placement important because the scanner must be able to ''see'' past the maker (e.g. it could not be placed against a wall).

III. REFLECTIVE FIDUCIAL SYSTEM

Our fiducial design consists of two highly reflective rectangles separated by a matte surface (Fig. 1). The reflective portions can be created using retro-reflective tape or any other material that reflects light in the infrared spectrum.

This shape was chosen because

- it is easy to manufacture,
- it results in spatially distinct clusters which can be used to help identify the beacon by comparing characteristics against the physical beacon, and
- its geometry facilitates the two-dimensional pose calculation.

The required size of the beacon scales with range and sensor resolution. Typically, the limiting factor is the vertical separation of the LiDAR scanlines. If all of the points on the beacon surface are collinear, a planar model cannot be fit to the data. For this reason, at least two scanlines must hit the beacon surface. With this requirement and the model shown in Fig. 2, the necessary beacon height h for a given range r

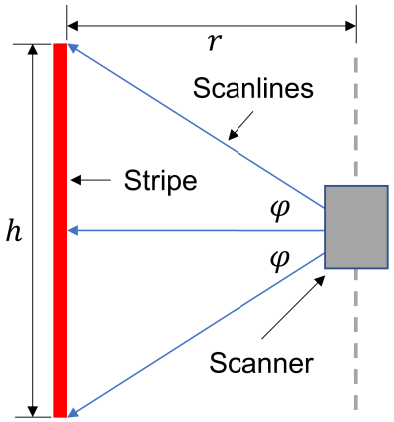

FIGURE 2. Scanner-beacon interaction model.

and a vertical scan line resolution of φ is

$$h = 2r\tan\varphi \tag{1}$$

and the required horizontal width w of the reflective rectangles is

$$w = 2r\tan(2\theta) \tag{2}$$

where θ is the horizontal angular resolution of the LiDAR scan points. The components of (2) are not included in Fig. 2, but the model is the same. Note that the model is designed to account for the worst case scenario of scanner-beacon alignment. For this reason, the equations are calculated using an additional scanline (i.e. the model includes three scanlines when only two are required) or horizontal return. This is because pitching the scanner causes different scanline alignments with the beacon. If scanlines are hitting at the top edge, middle, and bottom edge of the reflective stripe, then a slight pitch of the scanner will cause just two scanlines to hit the reflective surface.

Theoretically, the algorithm can succeed with as few as three total points hitting each reflective stripe on the beacon surface. However, experimentally this creates poor results because measurements from the LiDAR scanner are noisy and averaging more data points creates a higher accuracy localization. For this reason, we suggest designing the beacon to allow at least four points to hit the reflective surface per scanline (as in (2)).

Finally, the distance s between the inside edges of the reflective rectangles is given by

$$s = \frac{3}{2}\max(w, h) \tag{3}$$

By making the separation 50% larger than either dimension of the beacon stripe, the segmentation algorithm is easily able to create two distinct clusters from the beacon in the point cloud.

These values form only the lower limits that the algorithm requires to localize reliably. Increasing the size of the reflective rectangles while maintaining the required separation from (3) will increase the accuracy of the localization output because more scan points will hit the beacon surface. If a smaller beacon size is used, the system may still successfully localize in most of the region of interest. However, in this case, the system using the beacon-based localization should be designed to be robust to occasional failures.

A. SEGMENTATION

The beacon segmentation algorithm is responsible for segmenting the clusters of points which belong to the reflective beacon stripes (shown in Fig. 3). This object recognition problem is difficult because the algorithm must continue to operate as the sparsity of the point cloud grows with range. This means that the system must be able to successfully identify a cluster containing less than ten to over one thousand discrete LiDAR returns. We developed a custom segmentation pipeline which utilizes the specific geometric properties of the beacon to solve this problem. This algorithm represents the main contribution of this paper. It consists of the following steps:

1) intensity threshold filter;
2) voxel grid down-sample;
3) euclidean clustering with intensity threshold;
4) planar random sample consensus on individual clusters;
5) generation of identification heuristic parameters; and
6) selection of clusters based on identification heuristics.

1) INTENSITY THRESHOLD FILTER

The reflectivity of the beacon fiducials relative to the rest of the environment is the most obvious factor that differentiates the beacon in the point cloud data. For this reason, the first step in the segmentation process is to remove all points which are below the possible reflectivities of the beacon fiducial clusters. This spatially isolates the clusters belonging to the beacon while significantly reducing the dataset that needs to be processed by the next steps in the algorithm. LiDAR scanners typically have a documented intensity value which is indicative of a retro-reflective surface in the point cloud. This value can be used as the intensity threshold. If the system is expected to operate in exceptionally dusty environments, this threshold may need to be lower than the retro-reflective intensity value. This is because dust can accumulate on the beacon surface.

2) VOXEL GRID DOWN-SAMPLE

The number of points which hit the reflective stripes is highly dependent on the distance between the LiDAR scanner and the beacon. Changes in the point density can cause significant increases in the run-time of the algorithm. A voxel grid down-sample [20] is applied to each point cloud input to solve this problem. This down-sampling algorithm builds a euclidean grid in the 3D point cloud and finds the centroid of all points that fall within each grid cube. These centroids represent the down-sampled point cloud. This grid will reduce the number of points in the data set when the scanner is near the beacon while preserving the resolution when the scanner is far away.

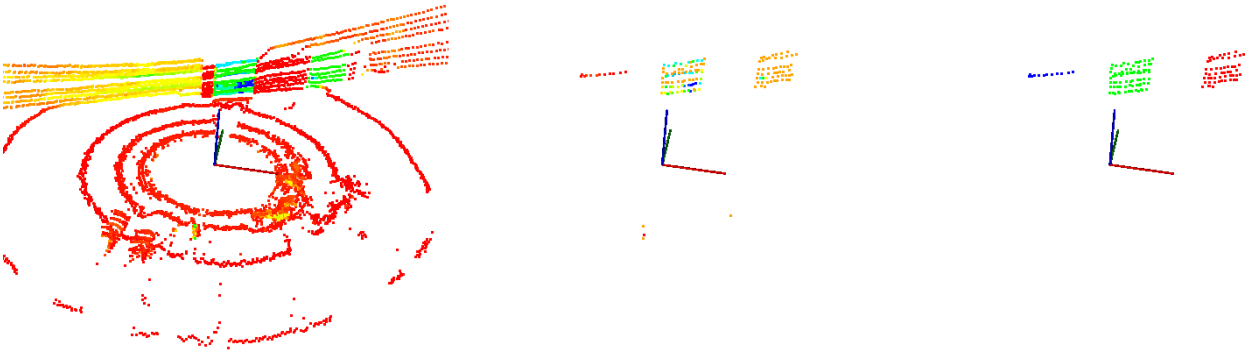

FIGURE 3. Reflective beacon in point cloud (left, color represents intensity), intensity filtered and down-sampled point cloud (middle, color represents intensity), and cluster extraction (right, color changes represent different clusters).

If the grid cube size is too large, the beacon may not be accurately identified in the point cloud by the next steps in the algorithm. However, as the voxel size decreases, the algorithm's total computing time increases. Therefore, the voxel size should be set to a small value and then increased until computational requirements are met while ensuring that the beacon can still be identified. We experimentally determined $0.1 \min(h, w)$ to be a good starting value, where the height h and width w of the beacon are calculated with (1) and (2).

3) EUCLIDEAN CLUSTERING

After the intensity threshold filter, the clusters belonging to the reflective beacon stripes will be spatially isolated. The segmentation algorithm then uses an intensity thresholded euclidean clustering algorithm to divide the point cloud into separate clusters. This algorithm grows clusters as long as a distance threshold l is met and an intensity similarity threshold is satisfied. A k-d tree [21] is used to efficiently search for spatially close points. An appropriate l threshold can be calculated with

$$l = \max(h, w) \tag{4}$$

to ensure all of the points from each cluster are accurately grouped. The intensity difference threshold i_d should be determined experimentally because this value is dependent on the particular LiDAR scanner used. The threshold should be set so that points that do not have similar reflectivity values are not grouped. However, the threshold should not be so low that points are erroneously rejected from the reflective stripe clusters. After completing the clustering operation, the algorithm prunes clusters by removing those which contain too few or too many points. The minimum value should be based on the sparsity of the generated point cloud at the maximum range of operation, while the maximum should be determined by the number of points in each cluster when the scanner is very close to the reflective beacon stripes. Three points can be used as an absolute minimum value, and a maximum of about 1000 is large enough for most scanners. These can be tuned by examining the density of the point clouds at the system's minimum and maximum operating ranges.

4) PLANAR RANDOM SAMPLE CONSENSUS

In rare circumstances, points representing obstructions near the reflective stripes can be fused with the beacon clusters. These erroneous points are rejected with a planar RANSAC algorithm [22]. If an adequate number of inliers are not found to fit a plane model in every cluster, the cluster is rejected. The exact outlier distance threshold is dependent on the accuracy of the 3D LiDAR scanner. We found that a value of two to three times the specified range accuracy of the LiDAR scanner reliably extracted the planar points in the cluster.

5) IDENTIFICATION METRIC GENERATION

The previous steps leave the segmentation process with a collection of point clusters that contain planar, highly reflective points. Now, the algorithm must determine which clusters belong to the beacon. The following metrics are used to make this determination:

- centroid separation;
- covariance matrix eigen values;
- angle between plane normal vectors; and
- difference in distance from cluster plane to origin.

After the clusters are partitioned from the point cloud, the centroid vector $\boldsymbol{c}$ to each cluster is computed by averaging the coordinates of each point. The euclidean distance c_s between any two cluster centroids can then be calculated with (5).

$$c_s = \|\boldsymbol{c}_1 - \boldsymbol{c}_2\| \tag{5}$$

The other heuristic parameters are derived with principal component analysis. First, the covariance matrix P of the cluster is calculated using

$$P = \frac{1}{n} \sum_{k=1}^{n} (\boldsymbol{x}_k - \boldsymbol{c}) (\boldsymbol{x}_k - \boldsymbol{c})^T \tag{6}$$

where n is the number of points in the cluster, $\boldsymbol{x}_k$ is the kth point in the cluster, and $\boldsymbol{c}$ is the respective cluster centroid. Next, eigen decomposition is performed on this covariance matrix to yield three eigen values (λ_1, λ_2, λ_3) and three associated eigen vectors ($\boldsymbol{v}_1$, $\boldsymbol{v}_2$, $\boldsymbol{v}_3$). These eigen values and

their associated eigen vectors are arranged such that $\lambda_1 > \lambda_2 > \lambda_3$. The eigen values of a covariance matrix are equal to the variance along the principal component axes. Therefore, after ordering, $\boldsymbol{v}_3$ points in the direction of minimum variance, which is the same direction as the least-squares fit normal vector of the planar cluster. The normal vector of a particular cluster will be referred to as $\boldsymbol{n}$ moving forward, where $\boldsymbol{n} = \boldsymbol{v}_3/||\boldsymbol{v}_3||$. Since the normal vectors are not unique, care must be taken to ensure that two anti-parallel normal vectors are reoriented to be parallel. In our implementation, all normal vectors are flipped to face the origin.

The angle between two cluster normal vectors $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ can be calculated with

$$\phi = \arccos\left(\boldsymbol{n}_1 \cdot \boldsymbol{n}_2\right). \tag{7}$$

The distance from a planar cluster to the origin can be calculated with

$$d = |\boldsymbol{n} \cdot \boldsymbol{x}| \tag{8}$$

where $\boldsymbol{x}$ is any point on the plane. Then, the origin distance difference o_d between two clusters can be calculated with

$$o_d = |d_1 - d_2|. \tag{9}$$

6) CLUSTER SELECTION

After generation of the cluster heuristic parameters, the algorithm has all of the necessary information to determine which clusters belong to the beacon.

First, the separation distance must satisfy

$$|c_s - (s + w)| < t_s \tag{10}$$

where s and w are calculated from (3) and (2). t_s represents the maximum allowable deviation from the correct centroid separation. This threshold can be set to a fraction of the expected distance between the centroids $s + w$. A value in the range of 20% to 40% worked well in our experiments.

Next, the Eigen values of the covariance matrix must satisfy $\lambda_1 > t_\lambda^1$, $\lambda_2 > t_\lambda^2$, and $\lambda_3 < t_\lambda^3$ (where t_λ^n is some threshold) to ensure that the cluster accurately fits a planar model. λ_3 must satisfy a maximum because this corresponds to the variance of the points along the normal vector. For a good quality plane fit, this variance should be low. If the cluster is planar, λ_1 and λ_2 should be large in comparison to λ_3 because the points will be distributed throughout the plane. As an example, a cluster consisting of only collinear points would be rejected because only one axis would have a large variance. These thresholds are best determined experimentally by calculating the eigen values of a correctly identified cluster. Finally, o_d must be less than a threshold t_{od} and $|\phi|$ must be less than t_ϕ. t_{od} sets the maximum allowable distance between the cluster planes (assuming they are parallel). The planes are parallel if t_ϕ is satisfied. Neither of these parameters is particularly sensitive, but there is some dependence on the scanner accuracy and beacon construction. t_ϕ is sensitive to warping of the beacon, which can occur if the beacon is constructed with flexible material. A value in the range of 20° to 30° worked well in our experiments. t_{od} is intended to reject clusters which do not lie in the same plane. It is also subject to beacon warping. For this reason, we suggest setting it to a percentage of $s + w$ (a larger beacon presents more possibility for warping and misalignment). We found that a value of 40% to 60% reliably rejected erroneous clusters while recognizing the beacon successfully.

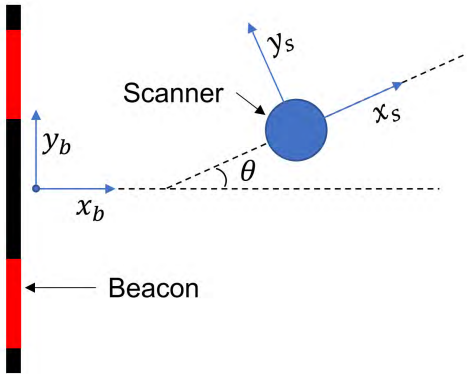

FIGURE 4. Coordinate systems.

If all of these conditions are satisfied, then the algorithm has found the set of clusters that belongs to the beacon to a high degree of certainty. In this case, the information generated in the identification phase can now be used in the localization step. However, if one of the heuristics is not satisfied, the algorithm cannot confidently use the cluster pair to generate a pose and will continue comparing clusters.

B. LOCALIZATION

After the beacon clusters have been successfully segmented, the identification algorithm has generated centroids ($\boldsymbol{c}_1$ and $\boldsymbol{c}_2$) and normal vectors ($\boldsymbol{n}_1$ and $\boldsymbol{n}_2$) of the matched cluster pair.

The localization algorithm uses this data to determine the 2D pose of the robot. First, the orientation of the clusters must be computed. The vectors $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ are the vectors drawn from the LiDAR sensor coordinate system origin to the centroid of each reflective rectangle in the LiDAR point cloud. The z-axis is oriented vertically (i.e. out of the page in Fig. 4) relative to the sensor, and the other axes are oriented according to Fig. 4. With these axes, the orientation can be determined by examining the sign of the angle difference of the two vectors. First, the beacon position vector $\boldsymbol{c}_b$ is computed with

$$c_b = \frac{\boldsymbol{c}_1 + \boldsymbol{c}_2}{2}. \tag{11}$$

Then, the angle of that vector in the xy-plane is computed with

$$\theta_b = \text{atan2}\left(c_{by}, c_{bx}\right) \tag{12}$$

and the angle of $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ with

$$\theta_1 = \text{atan2}\left(c_{1y}, c_{1x}\right) \tag{13}$$

and

$$\theta_2 = \text{atan2}\left(c_{2y}, c_{2x}\right). \tag{14}$$

Then, the left and right vectors $\boldsymbol{c}_l$ and $\boldsymbol{c}_r$ are determined by comparing θ_1 to θ_b. If $\theta_1 - \theta_b > 0$ (where care is taken to wrap angles to $(-\pi, \pi]$ in subtraction) then $\boldsymbol{c}_l = \boldsymbol{c}_1$ and $\boldsymbol{c}_r = \boldsymbol{c}_2$. Otherwise, $\boldsymbol{c}_l = \boldsymbol{c}_2$ and $\boldsymbol{c}_r = \boldsymbol{c}_1$.

Next, the centroid difference vector $\boldsymbol{c}_d$ is computed with

$$\boldsymbol{c}_d = \boldsymbol{c}_r - \boldsymbol{c}_l \tag{15}$$

and the beacon normal vector $\boldsymbol{n}_b$ with

$$\boldsymbol{n}_b = \frac{\boldsymbol{n}_1 + \boldsymbol{n}_2}{\|\boldsymbol{n}_1 + \boldsymbol{n}_2\|}. \tag{16}$$

After computation of these vectors, the 2D pose can be calculated. The xy-plane in the beacon coordinate system is defined as the plane that has a normal vector orthogonal to $\boldsymbol{c}_d$ and $\boldsymbol{n}_b$ while containing the point $\boldsymbol{c}_b$. The x-axis is parallel to $\boldsymbol{n}_b$ while the y-axis is parallel to $\boldsymbol{c}_d$. The x-coordinate can then be calculated with

$$x = -\boldsymbol{n}_b \cdot \boldsymbol{c}_b \tag{17}$$

since $\boldsymbol{n}_b$ is normalized, the y coordinate with

$$y = -\frac{\boldsymbol{c}_d \cdot \boldsymbol{c}_b}{\|\boldsymbol{c}_d\|} \tag{18}$$

and lastly θ with

$$\theta = \frac{\pi}{2} - \text{atan2}\left(c_{dy}, c_{dx}\right). \tag{19}$$

IV. EXPERIMENTAL RESULTS

We have presented an algorithm which segments a beacon from point cloud data and calculates a 2D localization. In this section, we prove the operation of this system through several experiments. First, we characterize the system accuracy and region of operation. Next, we verify the real-time operation of the algorithm on an embedded computer. Finally, the system is deployed on a prototype Martian mining robot.

Our algorithm was implemented in C++. The Point Cloud Library [23] provided pre-built implementations of some of the algorithms used in our system. We utilized a Velodyne VLP-16 3D LiDAR scanner. This sensor has 16 scan lines with a 2° vertical separation and better than 0.4° horizontal resolution. Its measurements are accurate to 3 cm and the scanner was configured to run at 10 Hz for all of the following tests. Due to the requirements of our deployment application, we chose to build a system that had a 6 m range. With this requirement, we arrived at the parameters shown in Table 1 to configure the algorithm and build the beacon. In this table, s_m is the maximum intensity value returned by the scanner. The reflective portions of the beacon were constructed using micro-prismatic retro-reflective tape laid on top of matte black foam board.

TABLE 1. Localization System Parameters.

Name	Description	Value
w	Beacon stripe width	0.360 m
h	Beacon stripe height	0.430 m
s	Beacon stripe separation	0.550 m
t_s	Cluster separation threshold	0.300 m
t_ϕ	Normal vector angle threshold	25°
t_λ^1,t_λ^2	Variance threshold 1, 2	5×10^{-4} m^2
t_λ^3	Variance threshold 3	1×10^{-4} m^2
t_{od}	Cluster plane separation	0.300 m
t_i	Intensity filter threshold	30% of s_m
i_d	Clustering intensity threshold	40% of s_m
t_r	RANSAC inlier threshold	0.080 m
v	Voxel cube size	0.050 m
l	Clustering length threshold	0.300 m

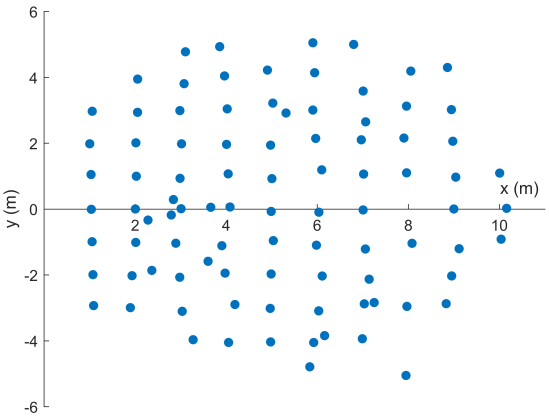

FIGURE 5. Localization test points.

A. ACCURACY AND REGION OF OPERATION

In order to characterize the accuracy of the system, we placed the scanner at 92 different points shown in Fig. 5 (beacon is at the origin). We then derived a ground truth pose estimate by triangulating the scanner pose using laser rangefinders. We used Bosch GLM 35 laser measures which have an accuracy of ±1.6 mm. This ground truthing method allows us to prove that our algorithm accurately calculates the pose of the scanner throughout the region of operation.

We triangulated the pose of the scanner by measuring a distance l_1 to the left edge of the beacon and l_2 to the right edge of the beacon from the center of our Velodyne scanner. Then, we calculate an internal angle ϕ_l of the formed triangle with

$$\phi_l = \arccos \frac{l_1^2 - u^2 - l_2^2}{-2ul_2} \tag{20}$$

where u is the distance from the left to the right edge of the beacon. If $\phi_l = \pi/2$ then the ground truth position can be calculated with

$$\begin{aligned} x_g &= l_2 \\ y_g &= -\frac{u}{2} \end{aligned} \tag{21}$$

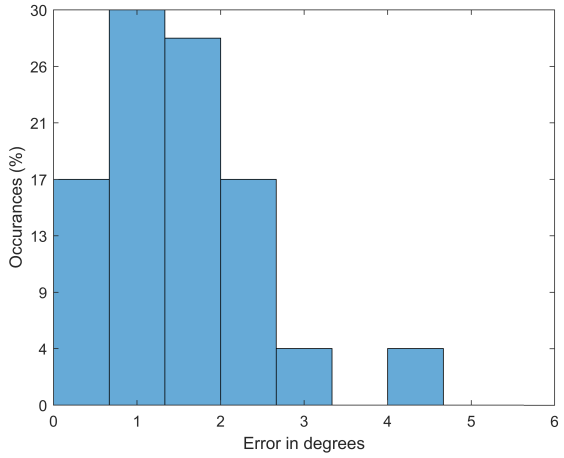

FIGURE 6. Histogram of yaw errors.

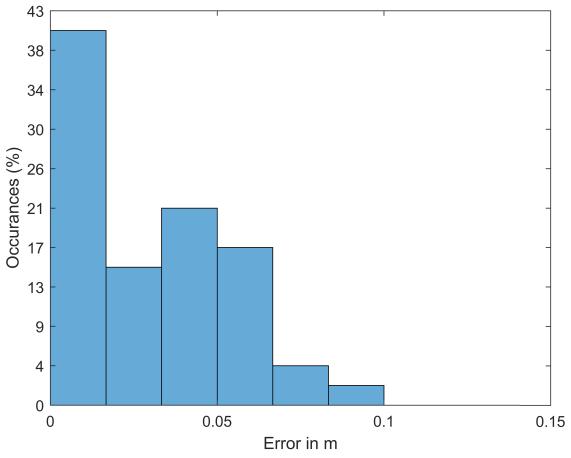

FIGURE 8. Histogram of errors along x-axis.

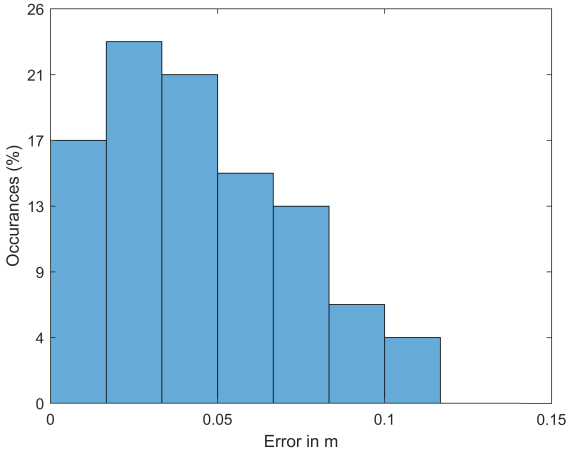

FIGURE 7. Histogram of euclidean distance errors.

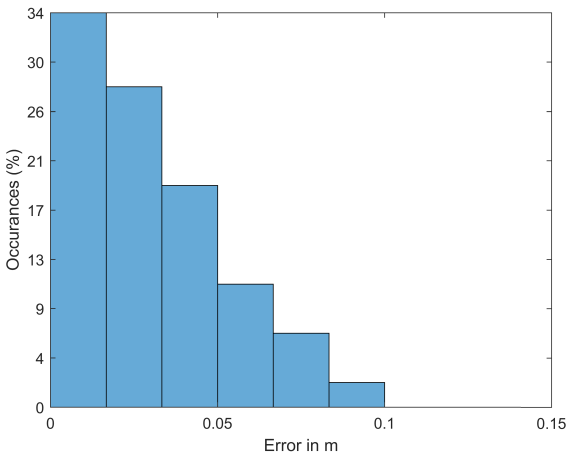

FIGURE 9. Histogram of errors along y-axis.

but if $\phi_l < \pi/2$ then

$$\begin{aligned} x_g &= l_2 \sin \phi_l \\ y_g &= -l_2 \cos \phi_l + \frac{u}{2} \end{aligned} \tag{22}$$

otherwise

$$\begin{aligned} x_g &= l_2 \sin (\pi - \phi_l) \\ y_g &= l_2 \cos (\pi - \phi_l) + \frac{u}{2}. \end{aligned} \tag{23}$$

The yaw angle can then be calculated by taking a second set of measurements at a point collinear with the scanner x-axis:

$$\theta_g = \text{atan2}\left(y_2^g - y_1^g, x_2^g - x_1^g\right) \tag{24}$$

The superscript indicates a ground truth point. We performed these measurements by attaching the scanner to a custom test fixture. This consisted of a rigid mount for the scanner and a long beam providing a surface to perform the secondary measurement to derive yaw.

Fig. 6 shows the histogram of yaw angle differences from the truth point (rangefinder triangulation) and the reflective beacon system measurements. A maximum error of 4.13° was observed over the measurements within the designed range. In Fig. 7, the euclidean distance error from the truth points to the algorithm measurements is shown. The maximum euclidean error across all measurements was 0.127 m. Fig. 8 and Fig. 9 show the histogram of errors along each axis. Note that our ground truth measurement range exceeds the designed range of the localization system. We tested at ranges of up to 10 m to demonstrate that our system gracefully degrades. During our data collection procedure, some localization failures were observed beyond the specified range of the system. Our histograms include only the errors from within the designed range (under 6 m). However, the algorithm never incorrectly identified the beacon even after exceeding the designed range, and the localization accuracy did not degrade significantly. This shows that our algorithm does not become unstable at the edge of its specified range. It can be safely used in a mobile robot which regularly enters and leaves the fiducial based localization system's area of operation with no special considerations. The maximum euclidean and angular errors outside the operational area were 0.249 m and 7.19°, respectively.

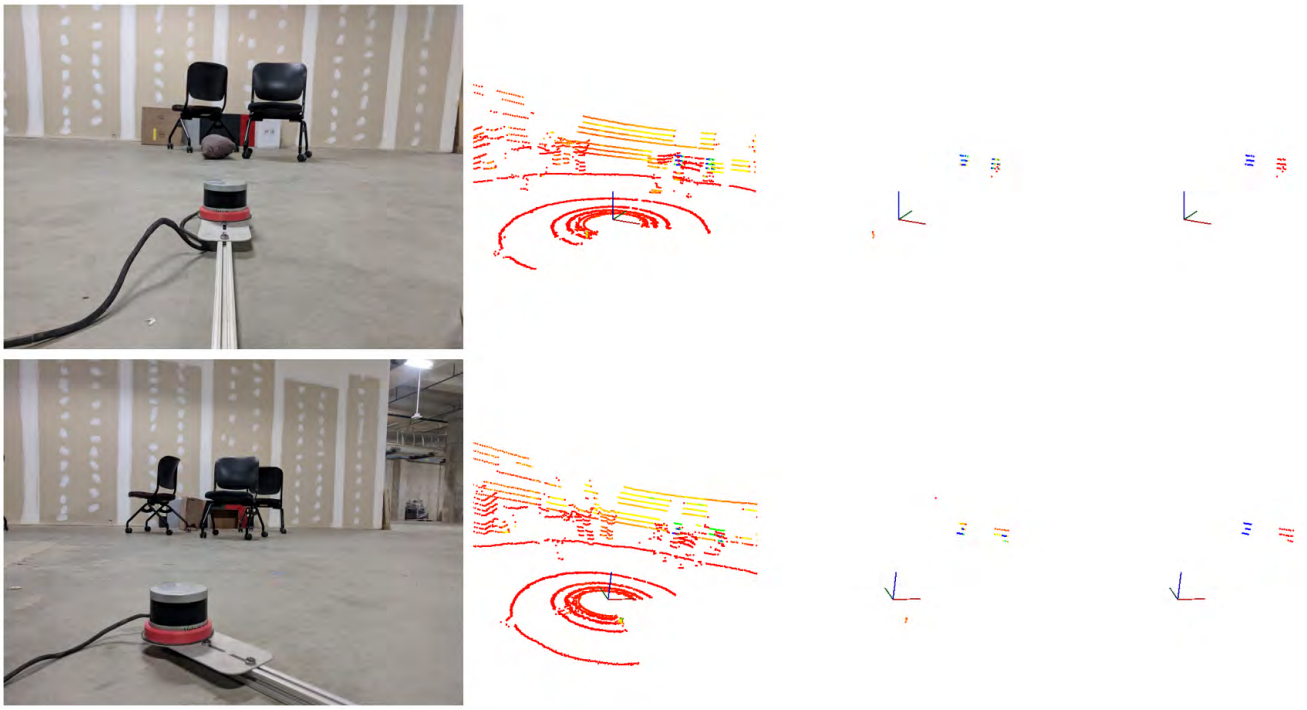

FIGURE 10. Highly occluded beacon view (left), point cloud view (middle-left), filtered point cloud (middle-right), cluster selection (right).

TABLE 2. Algorithm Performance.

Processor	Avg (ms)	Max (ms)	Min (ms)
VIA single board computer	0.815	2.875	0.404
Intel based laptop	0.378	1.431	0.182

B. REAL-TIME PERFORMANCE

The system was deployed on a VIA EPIA P910 Pico-ITX single board computer. This processor has an x86-64 architecture and a clock frequency of 1.2 GHz. We also tested the system on a laptop with a 2.50 GHz Intel i5-2520M processor. Both systems ran Ubuntu version 16.04 and the software was compiled using GCC version 6.3 with full optimizations. We ran the localization algorithm 163 times using point clouds from various ranges to account for different point densities. About 10% of the point clouds were failure cases (where the algorithm was unable to detect the beacon). The results are shown in Table 2.

This data proves that our algorithm is efficient and capable of operating in real-time while consuming a small percentage of the available computing power. Additionally, these results indicate that our system will continue to function in real-time with higher density point clouds, which will be important as higher resolution LiDAR scanners become readily available.

C. OCCLUSION RESISTANCE

Our system is highly resistant to occlusions. This is shown in Fig. 10, where the scanner is placed in front of two highly occluded beacons. In both test cases, the algorithm successfully identifies the beacon within the point cloud and localizes the scanner. We observed a maximum euclidean error of 0.044 m and a maximum yaw error of 3.1° across both examples. This shows that the center matte area of the beacon can be completely occluded with no effect on the localization output. The reflective regions can be occluded until the minimum number of points from each scanline are no longer able to reflect off the beacon surface. As shown in Fig. 10, this means that more than 50% of the stripe can be occluded with no ill-effects on the localization.

FIGURE 11. Robot navigating autonomously with localization system (beacon is against the back wall).

D. DEPLOYMENT

We deployed the system on a prototype Martian mining robot which competed in the NASA Robotic Mining Competition.

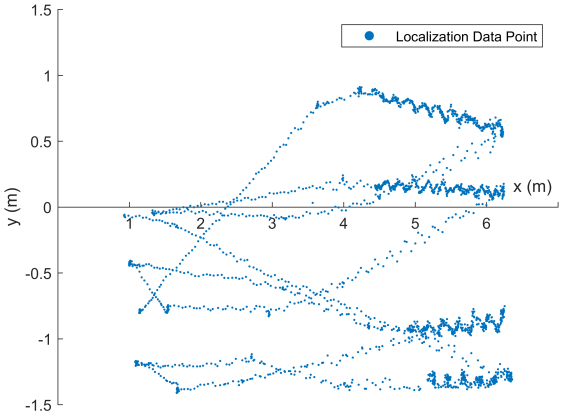

FIGURE 12. Robot localizations during a field deployment test.

Fig. 11 shows this system operating in a mining arena. A video of the robot navigating autonomously can be seen in [24]. During two years of operation, this localization system enabled every competition run to be completed fully autonomously. Fig. 12 shows the localizations collected from a deployment test. The different densities of points show changes in the robot's speed as it traverses across the simulated Martian landscape.

V. CONCLUSION

As shown by the experimental results, this paper presents an algorithm which robustly derives the two-dimensional position and orientation of a robot within a confined region. We showed that a LiDAR sensor based system has several distinct advantages over visual systems. In particular, direct depth measurement helps the algorithm be computationally efficient, resistant to beacon/marker occlusion, and invariant to lighting conditions. Unlike many other similar systems, our architecture requires the introduction of only a single artificial, passive marker into the environment. It has no dependence on environmental features beyond the beacon itself. In systems which consistently return back to a home location, this method introduces a simple way to eliminate errors induced by other localization methods such as wheel odometry. When the robot remains within a confined region, this system can provide a complete localization solution. High resolution LiDAR scanners are continuing to become more readily available. This trend will enable our system to operate with smaller beacon sizes at longer ranges, which widens its applicability.

REFERENCES

[1] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proc. IEEE Int. Conf. Robot. Automat.* Shanghai, China: IEEE, May 2011, pp. 3400–3407. doi: 10.1109/ICRA.2011.5979561.

[2] L. Sooyong and S. Jae-Bok, "Mobile robot localization using infrared light reflecting landmarks," in *Proc. Int. Conf. Control, Automat. Syst.* Seoul, South Korea: IEEE, 2007, pp. 674–677. doi: 10.1109/ICCAS.2007.4406984.

[3] M. Faessler, E. Mueggler, K. Schwabe, and D. Scaramuzza, "A monocular pose estimation system based on infrared LEDs," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*. Hong Kong: IEEE, May/Jun. 2014, pp. 907–913. doi: 10.1109/ICRA.2014.6906962.

[4] Y. Gu *et al.*, "Cataglyphis: An autonomous sample return rover," *J. Field Robot.*, vol. 35, no. 2, pp. 248–274, Mar. 2018. doi: 10.1002/rob.21737.

[5] R. A. Pratama and A. Ohya, "State estimation and control of an unmanned air vehicle from a ground-based 3D laser scanner," *J. Robot. Mechatron.*, vol. 28, no. 6, pp. 878–886, Dec. 2016. doi: 10.20965/jrm.2016.p0878.

[6] K. Shabalina, A. Sagitov, H. Li, and E. Magid, "Comparing fiducial marker systems occlusion resilience through a robot eye," in *Proc. 10th Int. Conf. Develop. eSyst. Eng. (DeSE)*. Paris, France: IEEE, Jun. 2017, pp. 273–278. doi: 10.1109/DeSE.2017.39.

[7] N. El-Sheimy, H. Hou, and X. Niu, "Analysis and modeling of inertial sensors using Allan variance," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 1, pp. 140–149, Jan. 2008. doi: 10.1109/TIM.2007.908635.

[8] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics* (Intelligent Robotics and Autonomous Agents). Cambridge, MA, USA: MIT Press, 2005.

[9] L. Kleeman, "Optimal estimation of position and heading for mobile robots using ultrasonic beacons and dead-reckoning," in *Proc. IEEE Int. Conf. Robot. Automat.* Nice, France: IEEE Comput. Soc. Press, May 1992, pp. 2582–2587. 10.1109/ROBOT.1992.220053.

[10] D. Kurth, G. Kantor, and S. Singh, "Experimental results in range-only localization with radio," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 1. Las Vegas, NV, USA: IEEE, Oct. 2003, pp. 974–979. doi: 10.1109/IROS.2003.1250754.

[11] C. D. McGillem and T. S. Rappaport, "A beacon navigation method for autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 38, no. 3, pp. 132–139, Aug. 1989. doi: 10.1109/25.45466.

[12] A. D. Buchan, E. Solowjow, D.-A. Duecker, and E. Kreuzer, "Low-cost monocular localization with active markers for micro autonomous underwater vehicles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*. Vancouver, BC, Canada: IEEE, Sep. 2017, pp. 4181–4188. doi: 10.1109/IROS.2017.8206279.

[13] A. Censi, J. Strubel, C. Brandli, T. Delbruck, and D. Scaramuzza, "Low-latency localization by active LED markers tracking using a dynamic vision sensor," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.* Tokyo, Japan: IEEE, Nov. 2013, pp. 891–898. doi: 10.1109/IROS.2013.6696456.

[14] H. Zhang, L. Zhang, and J. Dai, "Landmark-based localization for indoor mobile robots with stereo vision," in *Proc. 2nd Int. Conf. Intell. Syst. Design Eng. Appl.* Sanya, Hainan, China: IEEE, Jan. 2012, pp. 700–702. doi: 10.1109/ISdea.2012.640.

[15] S.-B. Han, J.-H. Kim, and H. Myung, "Landmark-based particle localization algorithm for mobile robots with a fish-eye vision system," *IEEE/ASME Trans. Mechatronics*, vol. 18, no. 6, pp. 1745–1756, Dec. 2013. doi: 10.1109/TMECH.2012.2213263.

[16] X. Zhong, Y. Zhou, and H. Liu, "Design and recognition of artificial landmarks for reliable indoor self-localization of mobile robots," *Int. J. Adv. Robotic Syst.*, vol. 14, no. 1, pp. 1–13, Jan. 2017. doi: 10.1177/1729881417693489.

[17] C. dos S. Fernandes, M. F. M. Campos, and L. Chaimowicz, "A low-cost localization system based on artificial landmarks," in *Proc. Brazilian Robot. Symp. Latin Amer. Robot. Symp.* Fortaleza, Brazil: IEEE, Oct. 2012, pp. 109–114. doi: 10.1109/SBR-LARS.2012.25.

[18] C.-L. Shih and Y.-T. Ku, "Image-based mobile robot guidance system by using artificial ceiling landmarks," *J. Comput. Commun.*, vol. 04, no. 11, pp. 1–14, 2016. doi: 10.4236/jcc.2016.411001.

[19] M. Fiala, "ARTag, a fiducial marker system using digital techniques," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2, Jun. 2005, pp. 590–596. doi: 10.1109/CVPR.2005.74.

[20] X.-F. Han, J. Jin, M.-J. Wang, W. Jiang, L. Gao, and L. Xiao, "A review of algorithms for filtering the 3D point cloud," *Signal Process., Image Commun.*, vol. 57, pp. 103–112, Sep. 2017. doi: 10.1016/j.image.2017.05.009.

[21] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975. doi: 10.1145/361002.361007.

[22] M. A. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981. doi: 10.1145/358669.358692.

[23] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Automat.* Shanghai, China: IEEE, May 2011, pp. 1–4. doi: 10.1109/ICRA.2011.5980567.

[24] (Jun. 2017). *NASA RMC 2017-Practice Run 1-Full Pit View*. [Online]. Available: https://youtu.be/UFzS0ZPrL7w

SPENCER DAVIS is currently pursuing the B.S. degree in electrical and computer engineering with The University of Alabama at Tuscaloosa, Tuscaloosa, AL, USA. He has worked on automation of large mining machines with Caterpillar Inc., and led the development of autonomy software for the Alabama Astrobotics Student Design Team. His research interests include computer vision, deep learning, real-time control, and robotics.

KENNETH G. RICKS received the B.S. degree in electrical engineering from The University of Alabama and the M.S. and Ph.D. degrees in computer engineering from The University of Alabama at Huntsville. He began his career as an Electronics/Computer Engineer with the Marshall Space Flight Center (MSFC), National Aeronautics and Space Administration (NASA), Huntsville, AL, USA, where he developed real-time simulations of various NASA vehicles and performed astronaut training in the Neutral Buoyancy Simulator.

In 2002, he joined the Department of Electrical and Computer Engineering, The University of Alabama at Tuscaloosa, as an Assistant Professor, where he currently serves as an Associate Professor and the Assistant Department Head. His research interests include embedded systems, parallel and real-time computation, robotics, and autonomous navigation.

RYAN A. TAYLOR received the Ph.D. degree in electrical and computer engineering from Mississippi State University, in 2018. He is currently an Assistant Professor with The University of Alabama at Tuscaloosa, Tuscaloosa, AL, USA. His research interests include remote sensing, asynchronous digital design, and engineering education.