LiDARTag: A Real-Time Fiducial Tag using Point Clouds

Jiunn-Kai Huang, Maani Ghaffari, Ross Hartley, Lu Gan, Ryan M. Eustice, and Jessy W. Grizzle

University of Michigan, Ann Arbor, MI 48109 USA,
{bjhuang, maanigj, rosshart, ganlu, eustice, grizzle}@umich.edu,
home page: https://www.biped.solutions

Abstract. Image-based fiducial markers are widely used in robotics and computer vision problems such as object tracking in cluttered or textureless environments, camera (and multi-sensor) calibration tasks, or vision-based simultaneous localization and mapping (SLAM). The state-of-the-art fiducial marker detection algorithms rely on consistency of the ambient lighting. This paper introduces LiDARTag, a novel fiducial tag design and detection algorithm suitable for light detection and ranging (LiDAR) point clouds. The proposed tag runs in real-time and can process data faster than the currently available LiDAR sensors frequencies. Due to the nature of the LiDAR's sensor, rapidly changing ambient lighting will not affect detection of a LiDARTag; hence, the proposed fiducial marker can operate in a completely dark environment. In addition, the LiDARTag nicely complements available visual fiducial markers as the tag design is compatible with available techniques, such as AprilTags, allowing for efficient multi-sensor fusion and calibration tasks. The experimental results, verified by a motion capture system, confirm the proposed technique can reliably provide a tag's pose and its unique ID code. All implementations are done in C++ and available at: https://github.com/brucejk/LiDARTag.

1 Introduction

Artificial landmarks referred to as *fiducial markers* are often designed for automatic detection via a specific type of sensor such as a camera (Wagner and Schmalstieg, 2003; Fiala, 2005a; Wang and Olson, 2016; DeGol et al, 2017). The marker usually consists of a *payload*, that is, a pattern that makes individual markers uniquely distinguishable, and a boundary surrounding the payload that is designed to assist with isolating the payload from its background. Such artificial landmarks have been successfully used in augmented reality (Wagner and Schmalstieg, 2003) and simultaneous localization and mapping (SLAM) (DeGol et al, 2018).

Images are sensitive to lighting variations, and therefore, visual fiducial markers heavily rely on the assumption of illumination consistency. As such, when lighting changes rapidly throughout a scene, the detection of visual markers can

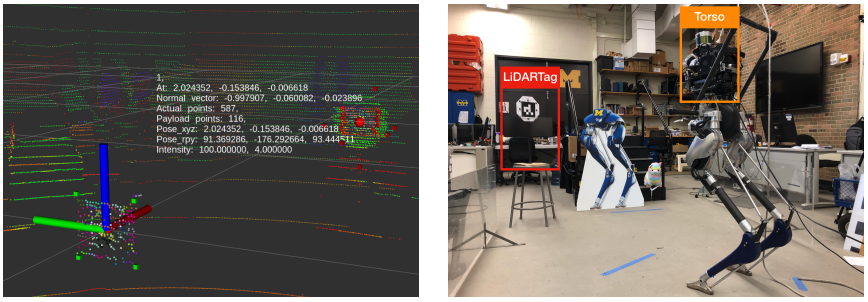


Fig. 1: This paper describes a real-time system that can detect fiducial markers in LiDAR point clouds, called LiDARTags. The fiducial markers can also be used with cameras, which could address an issue of images being sensitive to ambient lighting and provide a path to LiDAR-camera calibration. This figure shows a visualization of a LiDARTag in a full point cloud scan and the experimental setup.

fail. Alternatively, light detection and ranging devices (LiDARs) are robust to illumination changes due to the active nature of the sensor. In particular, rapid changes in the ambient lighting do not affect the detection of features in point clouds returned by a LiDAR. Unfortunately, LiDARs cannot detect the fiducial markers mentioned earlier. Hence, to utilize the advantages of both sensor modalities, a new type of fiducial marker that can be perceived by both LiDARs and cameras is required. Such a new marker can enable applications such as multi-sensor fusion and calibration tasks involving visual and LiDAR data.

In this paper, we propose a novel type of fiducial marker, called LiDARTag, as shown in Figure 1, that can be detected in real-time via both images and point clouds. In particular, the present work has the following contributions:

1. We develop a novel fiducial tag system, LiDARTag, that can be detected and decoded in real-time using both cameras and LiDAR point clouds.
2. We present performance evaluations of the LiDARTag against a vision-based baseline where a motion capture system provides ground truth data.
3. An implementation, using C++ and Robot Operating System (ROS) (Quigley et al, 2009), is available at: https://github.com/brucejk/LiDARTag.

The remainder of this paper is organized as follows. Section 2 presents a summary of the related work. Section 3 explains the tag design. Tag detection and decoding are discussed in Section 4. Experimental evaluations of the proposed LiDARTags is presented in Section 5. Finally, Section 6 concludes the paper and provides suggestions for future work.

2 Related Work

Fiducial marker systems were originally developed and used for augmented reality applications (Wagner and Schmalstieg, 2003) (Fiala, 2005a). More recently, fiducial markers have been widely used in the robotics community for object

detection and tracking, and for pose estimation (Klopschitz and Schmalstieg, 2007). Due to their uniqueness and fast detection rate, they are often used to improve Simultaneous localization and mapping (SLAM) systems as well (DeGol et al, 2018). Among the many popular camera-based fiducial markers are ARToolKit (Wagner and Schmalstieg, 2003), ARTag (Fiala, 2005a), AprilTag 1-3 (Olson, 2011) (Wang and Olson, 2016) (Krogius et al, 2018), and CALTag (Atcheson et al, 2010).

Currently, most fiducial markers are 2D and are captured with a camera. ARTag (Fiala, 2005a) (Fiala, 2005b) uses a 2D barcode to make decoding easier. AprilTag 1-3 (Olson, 2011) (Wang and Olson, 2016) (Krogius et al, 2018) introduced a lexicode-based (Trachtenbert, 1996) tag generation method in order to reduce false positive detection. ChromaTag (DeGol et al, 2017) proposes color gradients to speed up the detection process. RuneTag (Bergamasco et al, 2011) uses rings of dots to improve occlusion robustness and more accurate camera pose estimation. CCTag (Calvet et al, 2016) adopts a set of rings to enhance blur robustness. All of those detectors, however, only work on images.

Most of the methods for 3D object detection in point clouds using deep learning deploy a voxel grid representation (Song and Xiao, 2014) (Engelcke et al, 2017) (Song and Xiao, 2018). Recently, (Song and Xiao, 2016) (Engelcke et al, 2017) (Li, 2017) seek to to improve feature representation with 3D convolution networks, however, which require expensive computation. Similarly to proposed methods for 2D objects (Zitnick and Dollár, 2014) (Van de Sande et al, 2011) (Carreira and Sminchisescu, 2011), the proposed methods for 3D objects generate a set of 3D boxes in order to cover most of the objects in 3D space. However, none of these detectors or proposed methods has adequately addressed rotation, perspective transformations and domain adoption.

3 Tag Design and LiDAR Characteristics

This section describes some essential things to have in mind when designing and using a LiDAR-based tag system. In particular, it addresses how the unstructured point cloud from a LiDAR results in different considerations in the selection of a tag versus those used with a camera.

3.1 LiDAR Point Clouds vs Camera Images

Pixel arrays (i.e., an image) from a standard RGB-camera are very structured: the pixels are arranged in a uniform (planar) grid and each image consists of a fixed number of data points. A LiDAR returns (x, y, z) coordinates of an object relative to a fixed frame in the device as well as the intensity of the return signal. The resulting 3D point clouds are typically referred to as "unstructured" because:

– The number of returned points varies for each scan and for each beam. In particular, LiDAR returns are not uniformly distributed in angle or distance.

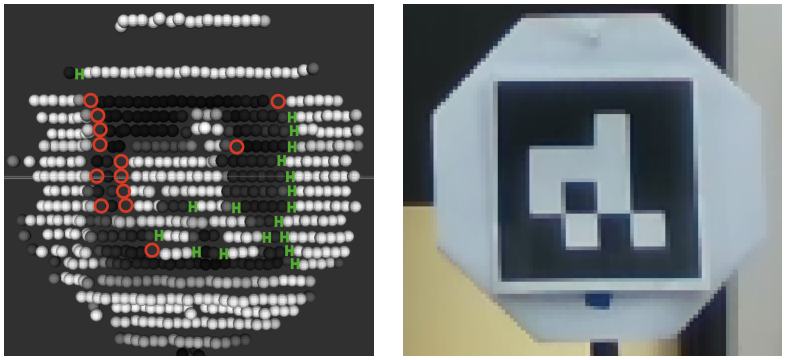


Fig. 2: This figure illustrates the unstructured nature of a LiDAR-point-cloud return (left) for a planar surface with black and white squares (right). The returns are more irregular at the black-white transitions. Red circles indicate missing returns and green bars highlight larger gaps between returned points.

- As shown in Fig. 2, high contrast between regions of a target's surface can result in missing returns and in varying spaces between returns.
- When used outdoors, the number of returned points is also influenced by environmental factors such as weather, especially temperature.

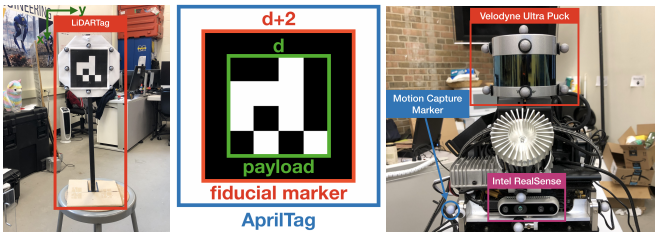
Consequently, as opposed to an image, there is not a fixed geometric relationship between the index numbers of returns from two different beams in a multi-beam LiDAR. A further difference is the density of the collected data; currently, basic cameras provide many more data points for a given surface size at a given distance than even high-end LiDARs. These summarized differences have an impact on how one approaches the design of a LiDAR-based tag system vs a camera-based tag system.

3.2 Tag Design

As mentioned in Sec. 3.1, a return from a LiDAR consists of an (x, y, z) measurement and an intensity value. It is proposed here to exploit the relative accuracy of a LiDAR's distance measurement to determine edge points when seeking to isolate a LiDARTag in a 3D point cloud. It is further proposed to use *intensity* to isolate and decode a tag's payload.

With these choices, a LiDARTag is assumed to consist of a planar fiducial marker rigidly attached to a freestanding 3D object as shown in Fig. 3a. In particular, in this paper, the fiducial marker cannot be attached to a wall. The fiducial marker is assumed to consist of a clearly delineated black and white boundary surrounding a payload consisting of a black and white pattern. The intensity difference of the LiDAR returns for the black and white boundary will be used in secondary edge detection, as described later in Sec. 4.1.

With these *assumptions*, tag families such as AprilTag 1-3 (Olson, 2011; Wang and Olson, 2016; Krogius et al, 2018), ARTag (Fiala, 2005a,b), or Inter-



(a) (b) (c)

Fig. 3: The left illustrates a possible LiDARTag consisting of two parts: a freestanding 3D object with a rigidly attached, planar fiducial marker. On the middle, a fiducial marker is outlined in red and the payload is outlined in green. In this particular marker, d is 4, the number of squares in a row.

Sense (Naimark and Foxlin, 2002) can be adapted for use in our LiDARTag system, but not those in CyberCode (Rekimoto and Ayatsuka, 2000), Cho et.al (Cho et al, 1998) nor CALTag (Atcheson et al, 2010).

For this initial study, we employ AprilTag 2 as our fiducial markers. Furthermore, within the AprilTag 2 family of markers, we select a tag16h5c5 tag, that is, a tag encoding 16 bits (i.e., 16 black or white squares), with a minimum Hamming distance of 5, and a complexity of 5. The Hamming distance measures the minimum number of bit changes (e.g., bit errors) required to transform one string of bits into the other. For example, the length-7 strings "1011111" and "1001011" have a Hamming distance of 2, whereas "1011110" and "1001010" have a Hamming distance of 3. The significance is that a lexicode with a minimum Hamming distance h can detect $h/2$ bit errors and correct up to $\lfloor (h-1)/2 \rfloor$ bit errors (Trachtenbert, 1996). The complexity of an AprilTag is defined as the number of rectangles required to generate the tags 2D pattern. For example, a solid pattern requires just one rectangle, whereas, a white-black-white stripe would need two rectangles (first draw a large white rectangle; then draw a smaller black rectangle). For further details, we refer reader to the coding discussion in Olson (Olson, 2011).

Remark 1. One may argue that other members of the AprilTags 2 family, such as tag49h15c15, tag36h11c10, tag36h10c10, and tag25h10c8, would be more appropriate. For example, including more bits tends to increase the number of distinct tags in a family, while a larger Hamming distance reduces false positives. However, for tags of the same physical size and the same distance from the sensor, more bits means fewer returns per square and a higher error rate for individual squares.

4 Tag Detection and Decoding

This section provides the individual steps for detecting a tag and decoding it. The overall pipeline is shown in Fig. 4.

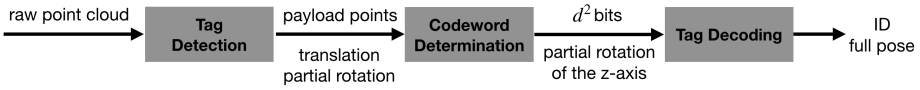


Fig. 4: The system contains three parts: tag detection, codeword determination and tag decoding. The detection step takes an entire LiDAR scan (up to 120,000 points from a Velodyne Ultra Puck) and outputs collections of likely payload points along with partial poses of the LiDARTags. Next, the tag's corners are identified and the LiDAR returns are associated to individual squares in the tag, with the weighting determined by nearness to a grid's center point. At this point we now have the 16 bits in the payload, though the rotation of the tag about a normal vector to the tag may be off by $\pm 90^\circ$ or 180°. The final step is to enter the four potential codewords (i.e., one 16-bit string for each of the four potential rotations of the tag) into a hash table to either decode the tag and determine its final rotation, or, falsify the codeword.

4.1 Tag Detection

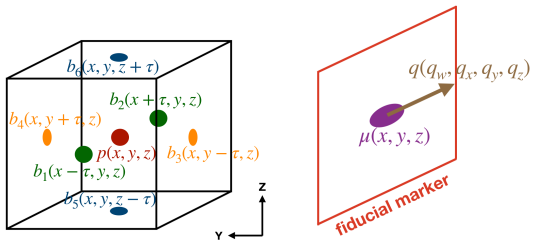
LiDARTag detection is the core of the LiDARTag system. To localize potential LiDARTags within the point cloud, the first step is to apply edge detection. The edge points are then grouped into distinct *clusters*. Most of these clusters will not contain tags. How the LiDAR returns are associated with each cluster is then explained in order to down select to a much smaller set of clusters that may contain valid LiDARTags. In Fig. 3b, to extract the fiducial marker rigidly attached to on a freestanding object, another edge detection is deployed. The last step, the partial pose (except the rotation about the z-axis) of the payload is estimated. The full rotation estimation will be addressed in Sec. 4.2 and Sec. 4.3.

Edge Detection As mentioned in Sec. 3.1, images are very structured in that the vertical and horizontal pixel-pixel correspondences are known. Consequently, various kinds of 2D kernels (Canny, 1987) can be applied for edge detection. However, unlike images, raw LiDAR point clouds are unstructured in that even if we have the indices of all points in each beam, we do not know the vertical point-point correspondence.

Consequently, we choose to use a 1D kernel to compute spatial gradients at each point to find edge points, which are defined as discontinuities in distance. Using distance gradients requires a freestanding LiDARTag from the background. Given the i^{th} point in the m^{th} beam at position $p^D_{i,m}$, the gradient of distance $|\nabla D(p_{i,m})|$ is defined as

$$|\nabla D(p_{i,m})| = max(||p^D_{i+l,m} - p^D_{i,m}||_2,\ ||p^D_{i-l,m} - p^D_{i,m}||_2). \tag{1}$$

where ℓ is a design choice (here, $\ell = 2$). If $\nabla D(p_{i,m})$ at a point exceeds a threshold, we then consider $p_{i,m}$ as a possible edge point. For speed in real time application, we do not apply further noise smoothing, edge enhancement, nor edge localization.



(a) (b)

Fig. 5: (a) shows a single edge point cluster. A cluster is defined as a cuboid in $\mathbb{R}^3$. When an edge point fails at linkage, a cluster will be created and it is centered at itself (x, y, z) with boundaries $(x \pm \tau, y \pm \tau, z \pm \tau)$, where τ is (tag size)/4. (b) shows a pose definition of a LiDARTag, which is fully defined by a fiducial marker on it.

Edge Clustering After determining the edge points in the current point cloud, we group them into clusters (cuboids in $\mathbb{R}^3$) using the single-linkage agglomerative hierarchical clustering algorithm (Johnson, 1967), with the linkage criteria chosen as signed Manhattan distance. Boundaries of a cluster are defined by the 6 center points of a cuboid's faces, as shown in Fig. 5a. The algorithm loops over each edge point $p^k = (x, y, z)$, either linking it to an existing cluster and updating its boundaries, or creating a new cluster. The boundaries $(b_1, \cdots, b_6)$ of a cluster are the maximum/minimum x, maximum/minimum y and maximum/minimum z among all edge points in the cluster. When a new cluster is created from an edge point, its boundaries are defined as $(x \pm \tau, y \pm \tau, z \pm \tau)$, with $\tau =$ (tag size)/4 as shown in Fig. 5a. The linkage criteria $L(p^k, c_j)$ between an edge point p^k and a cluster c_j is

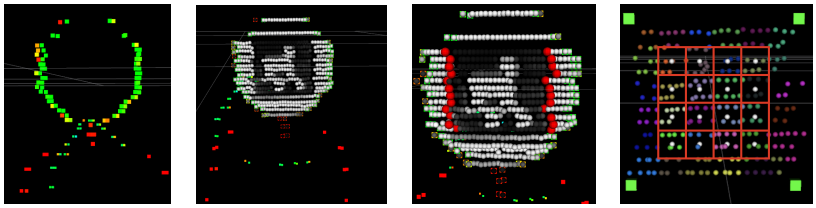
$$\forall i = 1, \cdots, 6 \qquad \min(b_i^k - \tau) \leq p^k \leq \max(b_i^k + \tau), \tag{2}$$

the signed Manhattan distance.

Cluster Validation and Fiducial Marker Extraction We have grouped the edge points into clusters as shown in Fig 6a. In practice, few (possibly none) of the clusters will contain a valid tag and thus it is important to be able to eliminate clusters that are clearly invalid. To do so, we first fill in LiDAR returns between the edge points of a cluster as shown in Fig. 6b. Two heuristics are used to validate a cluster: number of points η in a cluster, and number of edge points ψ of the payload *inside* the cluster. The values are determined by what type of tag family is chosen. Shown in Fig. 3b is a tag family which contains d^2 bits.

A lower bound on the number of points in a cluster is determined by the size of the fiducial marker. If the payload is $d \times d$, then the LiDARTag is roughly $(d+4) \times (d+4)$. If we assume a minimum of five returns for each bit in the tag, then the minimum number points in a valid cluster is

$$\eta \geq 5(d+4)^2. \tag{3}$$



(a) (b) (c) (d)

Fig. 6: Intermediate steps of the LiDARTag system. The system takes a full scan point cloud and applies edge detection as defined in (1) and associates the edge points into clusters as shown in (a). Using the edge points, the clusters are filled from the scan and as shown in (b). After checking that a cluster contains a minimum number of LiDAR returns to be valid, the boundaries of a candidate fiducial marker, as shown in (c), are determined from an intensity gradient, (4). The final steps are to associate the returns on the fiducial marker to individual grids of the tag, as illustrated in (d) and applying a weighted voting technique to determine color. The dots are the center of each grid shown by the red squares.

By construction of the LiDARTag, the boundaries of the payload can be detected by an intensity gradient,

$$|\nabla I(p_{i,m})| = max(|p^I_{i+\ell,m} - p^I_{i,m}|,\ |p^I_{i-\ell,m} - p^I_{i,m}|) \tag{4}$$

where ℓ is also a design choice (here, taken as one) and $p^I_{i,m}$ is the intensity value of the i^{th} point on the m^{th} beam. If $\nabla I(p_{i,m})$ exceeds a threshold, then $p_{i,m}$ is a payload edge point. To successfully decode a tag, we will need at least one beam on each row of the payload. Hence, the minimum number of payload edge points is

$$\psi \geq 2(d+2). \tag{5}$$

The above heuristics allow us to extract a potential fiducial marker from the LiDARTag through edge detection. The next step is to estimate the pose of the marker.

Partial Pose Estimation The coordinates of a LiDARTag are defined in Fig. 3. A convenient way of representing the pose of a LiDARTag is using a center point of the fiducial marker, denoted $\mu = (x, y, z)$, and a quaternion $q = (q_w, q_x, q_y, q_z)$ representing the rotation of the *fiducial marker* as shown in Fig. 5b. Given a set of N returns on a fiducial marker, the center of a LiDARTag is estimated as the mean of all the points p_i. The rotation of a LiDARTag, q, is defined with respect to a normal vector of the tag, which is estimated from the given N points via principal components analysis (PCA) (Price et al, 2006). The first two components of PCA are two vectors spanning the plane of the fiducial marker and the third component is the normal vector. The rotation of the tag about the normal vector is addressed in Sec. 4.2.

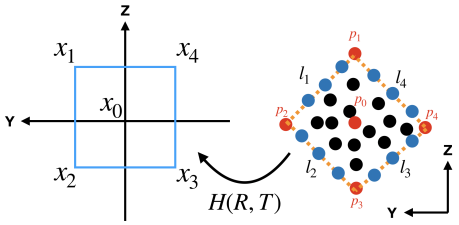


Fig. 7: The return points of a candidate fiducial marker are transformed to a square about the origin of the LiDAR's (y, z)-plane, see (6), before associating returns to individual grids.

4.2 Codeword Determination

To determine the codeword, each of the grids must be assigned a zero or a one. This is done by gridding the payload and voting, where a return point's weight is scaled by its distance from a grid's center point.

Building the Grid To make gridding easy, the return points of a candidate fiducial marker are mapped under a rigid body transformation, H, to a square about the origin of the LiDAR's (y, z)-plane, with sides of length equal to that of a tag. The transformation's translation vector T and a rotation matrix R, as illustrated in Fig. 7, are chosen to minimize the distance from the points $(x_1, \cdots, x_4)$ and $(p_1, \cdots, p_4)$. This will be reduced to a (constrained orthogonal) procrustes problem (Gower, 1975), namely a problem of the form

$$\Theta = \underset{\Omega:\ \Omega^T\Omega=I}{\arg\min}\ ||\Omega A - B||_F, \tag{6}$$

where for us, Ω will be a to-be-determined rotation matrix.

Without loss of generality, x_0 is assumed to be the origin, $(0, 0, 0)$. The translation T is thus given by

$$\begin{aligned} x_0 &= Rp_0 + T,\ \ x_0 = [0, 0, 0]^\mathsf{T} \\ T &= -Rp_0. \end{aligned} \tag{7}$$

The rest of the problem can be formulated as:

$$\|H\tilde{p}_i - x_i\|_2^2 = \left\| \begin{bmatrix} R & T \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} p_i \\ 1 \end{bmatrix} - \begin{bmatrix} x_i \\ 1 \end{bmatrix} \right\|_2^2 \tag{8}$$

$$= \|Rp_i + T - x_i\|_2^2 = \|Rp_i' - x_i\|_2^2\,, \tag{9}$$

$$\sum_{i=1}^{4} \|H\tilde{p}_i - x_i\|_2^2 = \sum_{i=1}^{4} \|Rp_i' - x_i\|_2^2 \tag{10}$$

$$= \left\| (Rp_1' - x_1) \vdots\ \cdots\ \vdots (Rp_4' - x_4) \right\|_F^2 \tag{11}$$

$$= \|R\mathbf{P} - \mathbf{X}\|_F^2\,. \tag{12}$$

where $\tilde{p_i} = [p_i\ 1]^\mathsf{T}$, $p_i' = p_i - p_0$, $\mathbf{P} = \begin{bmatrix} p_1' & p_2' & p_3' & p_4' \end{bmatrix}$, $\mathbf{X} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}$ and $\|\cdot\|_F$ is the Frobenius norm. The problem is then

$$R^* = \underset{R:\ R^T R = I}{\arg\min} \|R\mathbf{P} - \mathbf{X}\|_F^2 . \tag{13}$$

By the procrustes optimization problem (Gower, 1975), we have a closed form solution:

$$M = \mathbf{X}\mathbf{P}^\mathsf{T} = U\Sigma V^\mathsf{T} \tag{14}$$

$$R^* = UV^\mathsf{T}. \tag{15}$$

The rotation matrix R^* can be used to determine the orientation of the tag with respect to the normal vector, modulo 90° as discussed earlier; see Fig. 8. The modulo 90° ambiguity will be removed when decoding the tag; see Sec. 4.3.

Weighted Gaussian Voting to Assign Zeros and Ones After the transformation, the payload is gridded as in Fig. 6d, so that LiDAR returns can be associated with individual bits of the payload. After doing this, we found that doing the obvious thing, namely, assigning equal weights to each point in a grid, performs poorly. A weighted Gaussian voting technique was found to increase the correctness from roughly 0% to nearly 99%.

After gridding the extracted payload, we form d^2 multivariate Gaussian distributions

$$w_{p_i} \sim \mathcal{N}_k\left(\mu_k, \Sigma\right) \tag{16}$$

$$\mu_k = \begin{bmatrix} x_k \\ y_k \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix} \tag{17}$$

$$\sigma^2 = \frac{\text{size of fiducial marker}}{4(d+2)}, \tag{18}$$

where μ_k is the center of the k^{th} grid and the covariance is a constant determined by the size of a fiducial marker.

Given a point p_i in the k^{th} grid, its weight w_{p_i} determined by evaluating $\mathcal{N}_k\left(\mu_k, \Sigma\right)$. If there are N points in the a given grid, the possibility P_k of the k^{th} grid being white or black is estimated by

$$P_k = \frac{\sum_{i=1}^{N} w_{p_i} p_i^I}{\sum_{i=1}^{N} w_{p_i}}, \tag{19}$$

where p_i^I is the intensity value of the point p_i. Zero-one assignment is done by thresholding P_k.

4.3 Tag Decoding

Once the d^2 bits of a codeword have been determined, the decoding method is the same as for AprilTag (Olson, 2011; Wang and Olson, 2016). We have precomputed a hash table to store the codebook of this tag family. While looking

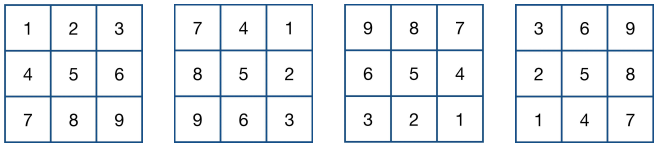


Fig. 8: Before decoding, the estimated rotation about the normal axis is only known modulo 90°, which means (a) to (d) yield the same normal vector. Accounting for the three possible rotations of $(90°, 180°, 270°)$, results in 4 possible codewords when looking up the correct id of a LiDARTag (only one of the four codewords is correct). From the correct codeword, the modulo 90° ambiguity is removed.

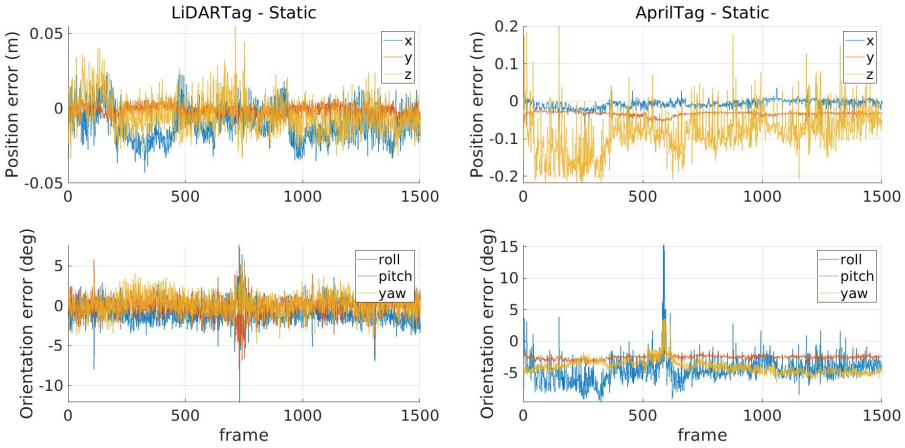


Fig. 9: Plots of estimation error for a static tag, LiDAR and camera.

up a codeword, it will be rotated three times as shown in Fig. 8 corresponding to rotations of $90°, 180°, 270°$ of the tag so we can fully recover the rotation of the payload about the normal vector. For more details, see (Olson, 2011).

5 Experimental Results

We now present experimental evaluations of the proposed LiDARTag. All experiments are done with a Velodyne Ultra Puck and an Intel RealSense camera as shown in Fig. 3c rigidly attached to the torso of a Cassie-series bipedal robot. The LiDARTag system runs on a laptop equipped with Intel® Core™ i7-7700HQ CPU @ 2.80 GHz. The dataset contains images (20 Hz) and scans of point clouds (10 Hz). We use the Robot Operating System (ROS) (Quigley et al, 2009) to communicate between sensors. The motion capture system developed by Qualisys is used as a proxy for ground truth poses. The setup consists of 18 motion capture cameras with markers attached to tags, a LiDAR and a camera.

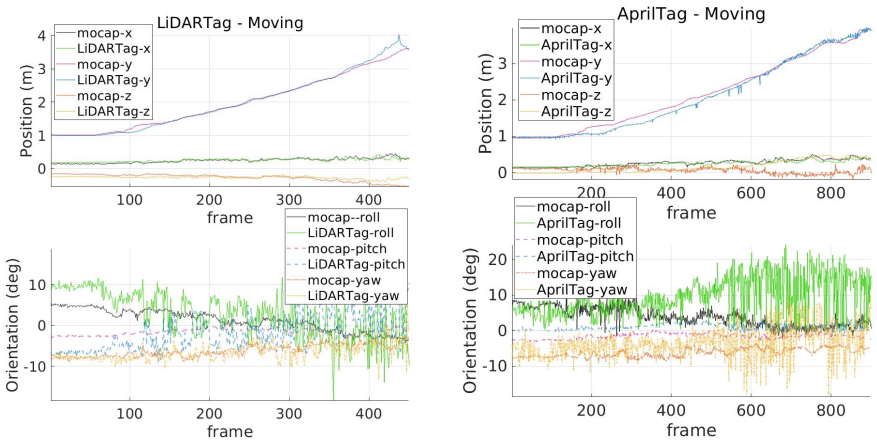


Fig. 10: Estimated position and orientation when the LiDAR and camera are moving and the tag is stationary.

5.1 Computational Time and Decoding Accuracy

In this work, we choose a stop sign as our freestanding 3D object and AprilTag 2 as our fiducial markers, as shown in Fig. 3. To evaluate the computation time and decoding accuracy, we placed the torso at 2 meters away from a LiDARTag for both indoor and outdoor experiments. The computation time of each step of the pipeline is shown in Table 1. Indoors, we have more edge points due to a cluttered environment. However, we have fewer clusters indoor because detected edge points are closer to each other resulting in many of them being clustered together. The computation time in an outdoor environment was slower than indoor because we have more clusters to create and validate. In both environments, the system achieves real-time performance (at least 20 Hz). The decoding accuracy in both environments is 99%.

In the indoor and outdoor environments, the equal-weight decoding method fails to determine a codeword for 0.4152% and 0.0822% of the time; codeword are determined but the decoder cannot find a corresponding codeword from the codebook for 85.6055% and 84.4984% of the time; the decoder finds a wrong codeword for 13.9792% and 15.4194% of the time. All together, this amounts to 0.0001% and 0.0000% accuracy, indoor and outdoor, respectively.

The above work used tag16h5. Prior to settling on it, we also tested other tag families in AprilTag 2, such as tag49h14 and tag25h10, with lower accuracy, due to the sparsity of our point clouds. If the number of points in a grid, after the remapping in Sec. 4.2, is less than 5, it is considered a bad bit. Since the tag16h5 has a Hamming distance of 5, we allow up to 2 bad bits in a payload. If more than 2 bad bits are detected, the payload was rejected.

Table 1: LiDARTag Accuracy and Computation Time. Unless it is mentioned, all times are reported in milliseconds. The number of points, number of edge points and number of clusters are the average of all scans. The decoding accuracy of the weighted Gaussian voting technique in both environments is 99%. Most of the computation time is spent on clustering edge points. Actual time spent on validation and other processes is relatively small. The accuracy of equal weight method is poor due to failed determination of a codeword, no existing codewords found in the codebook, or finding a wrong codeword.

Indoor

Collecting time	No. Scans	No. Points	No. Edges	No. Clusters	Accuracy of Equal Weight	Accuracy of Gaussian Weight
120 sec	1240	53548	3813	779	0.0001%	99.6371%
Clustering	Validation	Extraction	Normal Vec.	Decoding	Total computation time	
24.004	1.754	0.0271	0.0339	1.199	29.496896 (33.9 Hz)	

Outdoor

Collecting time	No. Scans	No. Points	No. Edges	No. Clusters	Accuracy of Equal Weight	Accuracy of Gaussian Weight
266 sec	3820	52477	2416	1757	0.0000%	99.7382%
Clustering	Validation	Extraction	Normal Vec.	Decoding	Total computation time	
38.096	4.741	0.0290	0.0319	1.202	44.655821 (22.4 Hz)	

In our work, we printed the AprilTags on A4 paper. Users could also print larger tags and use a larger Hamming distance to allow more bad bits, if the 2-bit limitation appears as a critical issue.

5.2 LiDARTag Pose Accuracy Evaluation

As described in Sec. 4.1, 4.2 and shown in Fig. 5b, the LiDARTag pose (μ and q) is defined by the middle of a fiducial marker and the normal vector of a fiducial marker, respectively. To evaluate pose accuracy of the proposed LiDARTag, we did two experiments. In the first experiment, we held Cassie's torso in a fixed position, and recorded for 2 minutes. The results are shown in Fig. 9.

In the second experiment, we moved the torso slowly back and forth as well as rotating it. In the Velodyne driver, it combines several *packets*, a small piece of point cloud, into a full scan of point cloud. Therefore, instant jumps will cause motion distortion effect. However, it could be improved by motion compensation or applying a filter. The result is shown in Fig. 10.

5.3 Discussion and Limitations

In this work, we use a Velodyne 32-beam Ultra Pack. The range limitation is because the LiDARTag has to be within a sufficiently *dense* area of the LiDAR's returns. While a typical LiDARs' vertical field of view (FOV) is 40°, only 20° of the FOV will have densely spaced beams. If a LiDARTag is not within the dense area of a LiDAR, it will not satisfy requirement of (5), resulting in failed

detections. As shown in Table 1, most of the computation time is spent on clustering edge points. Actual time spent on validation and other processes is relatively small. Thus, the number of LiDARTags in the FOV has little effect on computation time.

As mentioned in Sec. 5.2, a full scan (point cloud) from a Velodyne consists of several packets. Motion compensation is required if a robot moves quickly. One solution is to transform each packet to the state estimation frame, correct the motion distortion effect, and transform the packet back to the sensor frame. For Cassie Blue, we send each individual packet to the IMU frame and use an invariant extended Kalman filter (Barrau and Bonnabel, 2016; Hartley et al, 2018a,b) to compensate the motion distortion. The filter runs at 2 kHz and compensates LiDAR packets without any noticeable time delays.

As summarized in Sec. 3.1, the unstructured nature of a LiDAR's point cloud (irregular point distribution, missing returns, and varying spaces between returns) and the overall sparsity of points on a given marker made the decoding process challenging. In this paper, we proposed several means of meeting these challenges. One may suspect that "deep learning" could obviate many of the discrete steps we introduced. We gave this a try ourselves but we were unable to overcome issues such as rotation invariance of the network to tag orientating and domain adoption, which led us to employ the proposed remapping and weighted Gaussian voting to solve this problem. However, in the future we are interested in exploring the application of deep learning for this problem in more depth.

6 Conclusion and Future Work

We presented a novel fiducial tag system specifically for point clouds. The developed fiducial tag system runs in real-time (at 30 Hz) while it can handle a full scan of raw point cloud from the employed Velodyne Ultra Pack (up to 120,000 points per scan). The presented fiducial tags can also be used with cameras, suggesting its potential application for solving the camera-LiDARcalibration problem as the next step.

In the future, we shall use the developed LiDARTag within SLAM and calibration frameworks. We are currently working on adopting the long short-term memory (LSTM) (Luo et al, 2018; Xiang and Fox, 2017) to improve the decoding accuracy by incorporating the time-correlation information and rotation invariant networks (Marcos et al, 2017; Shi et al, 2018; Cheng et al, 2016; Worrall et al, 2017). Furthermore, in order to bring continuity to the point query result, Gaussian processes (Rasmussen and Williams, 2006) could be used in the decoding process if the number of points of a payload is small.

Acknowledgment

Funding for this work was provided by the Toyota Research Institute (TRI) under award number N021515. Funding for J. Grizzle was in part provided by TRI and in part by NSF Award No. 1808051. This article solely reflects the opinions and conclusions of its authors and not the funding entities.

Bibliography

Atcheson B, Heide F, Heidrich W (2010) Caltag: High precision fiducial markers for camera calibration. In: VMV, Citeseer, vol 10, pp 41–48

Barrau A, Bonnabel S (2016) The invariant extended kalman filter as a stable observer. IEEE Transactions on Automatic Control 62(4):1797–1812

Bergamasco F, Albarelli A, Rodola E, Torsello A (2011) Rune-tag: A high accuracy fiducial marker with strong occlusion resilience. In: Proc. IEEE Conf. Comput. Vis. Pattern Recog., IEEE, pp 113–120

Calvet L, Gurdjos P, Griwodz C, Gasparini S (2016) Detection and accurate localization of circular fiducials under highly challenging conditions. In: Proc. IEEE Conf. Comput. Vis. Pattern Recog., pp 562–570

Canny J (1987) A computational approach to edge detection. In: Readings in computer vision, Elsevier, pp 184–203

Carreira J, Sminchisescu C (2011) Cpmc: Automatic object segmentation using constrained parametric min-cuts. IEEE Transactions on pattern analysis and machine intelligence 34(7):1312–1328

Cheng G, Zhou P, Han J (2016) Rifd-cnn: Rotation-invariant and fisher discriminative convolutional neural networks for object detection. In: Proc. IEEE Conf. Comput. Vis. Pattern Recog., pp 2884–2893

Cho Y, Lee J, Neumann U (1998) A multi-ring color fiducial system and an intensity-invariant detection method for scalable fiducial-tracking augmented reality. In: In IWAR, Citeseer

DeGol J, Bretl T, Hoiem D (2017) Chromatag: a colored marker and fast detection algorithm. In: Proc. IEEE Conf. Comput. Vis. Pattern Recog., pp 1472–1481

DeGol J, Bretl T, Hoiem D (2018) Improved structure from motion using fiducial marker matching. In: Proc. European Conf. Comput. Vis., pp 273–288

Engelcke M, Rao D, Wang DZ, Tong CH, Posner I (2017) Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In: Proc. IEEE Int. Conf. Robot. and Automation, IEEE, pp 1355–1361

Fiala M (2005a) Artag, a fiducial marker system using digital techniques. In: Proc. IEEE Conf. Comput. Vis. Pattern Recog., IEEE, vol 2, pp 590–596

Fiala M (2005b) Comparing artag and artoolkit plus fiducial marker systems. In: IEEE International Workshop on Haptic Audio Visual Environments and their Applications, IEEE, pp 6–pp

Gower JC (1975) Generalized procrustes analysis. Psychometrika 40(1):33–51

Hartley R, Jadidi MG, Grizzle J, Eustice RM (2018a) Contact-aided invariant extended kalman filtering for legged robot state estimation. In: Proceedings of Robotics: Science and Systems, Pittsburgh, Pennsylvania

Hartley R, Jadidi MG, Grizzle JW, Eustice RM (2018b) Contact-aided invariant extended kalman filtering for legged robot state estimation. arXiv preprint arXiv:180510410

Johnson SC (1967) Hierarchical clustering schemes. Psychometrika 32(3):241–254

Klopschitz M, Schmalstieg D (2007) Automatic reconstruction of wide-area fiducial marker models. In: 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, IEEE, pp 71–74

Krogius M, Haggenmiller A, Olson E (2018) Flexible layouts for fiducial tags

Li B (2017) 3d fully convolutional network for vehicle detection in point cloud. In: Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst., IEEE, pp 1513–1518

Luo Y, Ren J, Wang Z, Sun W, Pan J, Liu J, Pang J, Lin L (2018) Lstm pose machines. In: Proc. IEEE Conf. Comput. Vis. Pattern Recog., pp 5207–5215

Marcos D, Volpi M, Komodakis N, Tuia D (2017) Rotation equivariant vector field networks. In: Proc. IEEE Int. Conf. Comput. Vis., pp 5048–5057

Naimark L, Foxlin E (2002) Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In: Proceedings of the 1st International Symposium on Mixed and Augmented Reality, IEEE Computer Society, p 27

Olson E (2011) Apriltag: A robust and flexible visual fiducial system. In: Proc. IEEE Int. Conf. Robot. and Automation, IEEE, pp 3400–3407

Price AL, Patterson NJ, Plenge RM, Weinblatt ME, Shadick NA, Reich D (2006) Principal components analysis corrects for stratification in genome-wide association studies. Nature genetics 38(8):904

Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, Wheeler R, Ng AY (2009) ROS: an open-source Robot Operating System. In: ICRA workshop on open source software

Rasmussen C, Williams C (2006) Gaussian processes for machine learning, vol 1. MIT press

Rekimoto J, Ayatsuka Y (2000) Cybercode: designing augmented reality environments with visual tags. In: Proceedings of DARE 2000 on Designing augmented reality environments, ACM, pp 1–10

Van de Sande KE, Uijlings JR, Gevers T, Smeulders AW, et al (2011) Segmentation as selective search for object recognition. In: Proc. IEEE Int. Conf. Comput. Vis., vol 1, p 7

Shi X, Shan S, Kan M, Wu S, Chen X (2018) Real-time rotation-invariant face detection with progressive calibration networks. In: Proc. IEEE Conf. Comput. Vis. Pattern Recog., pp 2295–2303

Song S, Xiao J (2014) Sliding shapes for 3d object detection in depth images. In: Proc. European Conf. Comput. Vis., Springer, pp 634–651

Song S, Xiao J (2016) Deep sliding shapes for amodal 3d object detection in rgb-d images. In: Proc. IEEE Conf. Comput. Vis. Pattern Recog., pp 808–816

Song S, Xiao J (2018) Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proc. IEEE Conf. Comput. Vis. Pattern Recog., pp 4490–4499

Trachtenbert A (1996) Computational methods in coding theory. Master's thesis, University of Illinois at Urbana-Champaign

Wagner D, Schmalstieg D (2003) Artoolkit on the pocketpc platform. In: 2003 IEEE International Augmented Reality Toolkit Workshop, IEEE, pp 14–15

Wang J, Olson E (2016) Apriltag 2: Efficient and robust fiducial detection. In: Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst., IEEE, pp 4193–4198

Worrall DE, Garbin SJ, Turmukhambetov D, Brostow GJ (2017) Harmonic networks: Deep translation and rotation equivariance. In: Proc. IEEE Conf. Comput. Vis. Pattern Recog., pp 5028–5037

Xiang Y, Fox D (2017) Da-rnn: Semantic mapping with data associated recurrent neural networks. arXiv preprint arXiv:170303098

Zitnick CL, Dollár P (2014) Edge boxes: Locating object proposals from edges. In: Proc. European Conf. Comput. Vis., Springer, pp 391–405