

# Unsupervised Calibration for Multi-beam Lasers

Jesse Levinson and Sebastian Thrun

## 1 Introduction

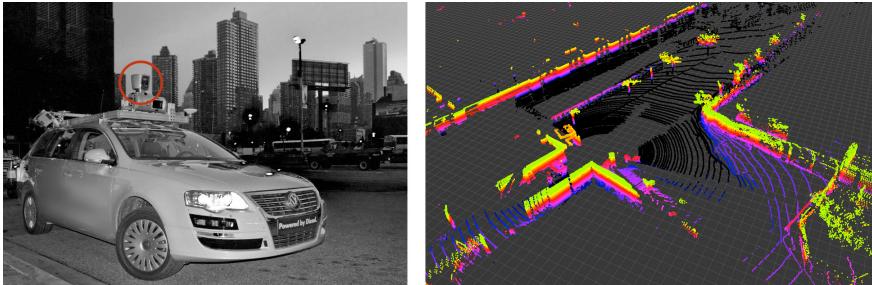
Light Detection and Ranging (LIDAR) sensors have become increasingly common in both industrial and robotic applications. LIDAR sensors are particularly desirable for their direct distance measurements and high accuracy, but traditionally have been configured with only a single rotating beam. However, recent technological progress has spawned a new generation of LIDAR sensors equipped with many simultaneous rotating beams at varying angles, providing at least an order of magnitude more data than single-beam LIDARs and enabling new applications in mapping [6], object detection and recognition [15], scene understanding [16], and SLAM [9].

In order to effectively harness this massive increase in beam count, new calibration methods are required. First, calibrating angles and range readings for tens or hundreds of beams is a substantially harder problem than calibrating one or a few beams. Second, for applications that utilize the intensity returns of LIDAR sensors, e.g. recent mapping and localization work [6], it is important that the intensity remittance values agree across beams. In both cases, the number of parameters makes supervised measuring and calibration at best tedious and at worst infeasible.

To date, there has been limited research on supervised calibration for multi-beam LIDARs. The most popular such unit as of this writing is the Velodyne HD-64E spinning LIDAR, which has been used extensively for many recent robotics applications. A representative scan from such a sensor is shown in Fig. 1 (right). In [4], the authors present a supervised calibration technique for this LIDAR requiring a dedicated calibration target and many hand measurements, followed by a traditional optimization step. Indeed, the manufacturers of this laser built a dedicated calibration facility which they use to collect thousands of measurements followed by an unpublished optimization routine, in order to provide a calibration of each beam to the customer.

---

Jesse Levinson · Sebastian Thrun  
Stanford University, 353 Serra Mall Stanford CA 94305  
e-mail: jessel@stanford.edu, thrun@stanford.edu



**Fig. 1** The Velodyne LIDAR. The sensor is mounted on the roof of our vehicle, circled in red (left). It returns 360-degree scans from its 64 beams (right). Points are colored by height for visual clarity.

In the case of single-beam LIDARs, there have been attempts at unsupervised recovery of roll, pitch, and yaw in a known rectangular enclosure [2] as well as an attempt to estimate the noise parameters of a single-beam LIDAR [3]. Recent work has provided algorithms for calibrating one [10] or two [1] single-beam LIDARs on a moving vehicle platform using hand-placed retroreflective calibration targets, which additionally requires an intensity threshold for correspondences.

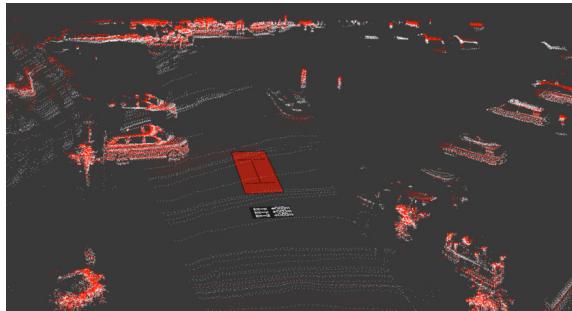
None of these techniques extends to the unsupervised calibration of a multi-beam LIDAR. Indeed, we are unaware of any algorithm in the literature to date that is able to recover the extrinsic pose of any LIDAR unit relative to a vehicle frame when it is the only such sensor on the vehicle and when the environment has no particular known features. In addition, we are similarly unaware of an existing algorithm that calibrates individual beam parameters for multi-beam units without hand-measured environmental features.

Although hand measurements are practical in some cases for single-beam sensors, especially for translational offsets, it is particularly difficult to measure sensor orientation with high accuracy in the absence of a dedicated calibration environment. Furthermore, for sensors with many beams, such measurements may take prohibitively long and would inevitably be suboptimal in accuracy. Finally, techniques that require specifically placed retroreflective targets with careful thresholding to pick them apart from the rest of the environment are not applicable in all settings a robot might encounter; they also result in the consideration of only a tiny fraction of the available data.

Thus, we propose a novel fully unsupervised extrinsic and intrinsic calibration method for multi-beam LIDAR sensors that requires no calibration target, no labeling, and no manual measurements. Given a multi-beam LIDAR attached to a moving platform or robotic vehicle along with accompanying inertial measurement unit (IMU) readings, our algorithm computes hundreds of sensor parameters from only seconds of data collected in an arbitrary environment without a map.

Our contribution consists of three complimentary unsupervised calibration algorithms. The first discovers the LIDAR's extrinsic 6-dimensional pose relative to the vehicle's inertial frame, including translation and rotation. The second estimates

**Fig. 2** Velodyne points from two adjacent beams (of 64) accumulated over time and projected into 3D; one beam colored red, the other white. Due to known vehicle motion, both beams tend to see the same surfaces.



optimal vertical and horizontal angles for each individual beam and an additive distance offset for each beam’s range reading. Finally, the third derives a fully Bayesian generative model for each beam’s remittance intensity response to varying surface reflectivities in the environment.

In the sections that follow, we will describe each of the above algorithms conceptually. We will then provide details about our particular implementation of these techniques on a ground vehicle with a roof-mounted 64-beam rotating LIDAR, along with several results demonstrating the effectiveness of these algorithms even when presented with poor initial calibrations. Finally, we will discuss implications for related applications and possible extensions for future research.

## 2 Extrinsic Calibration

In the case of a multi-beam LIDAR, extrinsic calibration considers the mounting location of the entire unit relative to the vehicle’s own coordinate frame, while intrinsic calibration considers the configuration of each individual beam inside the unit. In this section we present a method for extrinsic calibration, assuming a known intrinsic calibration.<sup>1</sup>

At the most basic level, our approach for both calibrations leverages the simple observation that laser returns projected into three dimensions are not randomly distributed in space. Indeed, because the returned points are reflections off of physical surfaces, it is impossible for a properly calibrated sensor traveling a known trajectory to return a collection of accumulated points that is randomly distributed in three dimensions. As such, the proposed method relies only on the weak assumption that points in space tend to lie on contiguous surfaces.

Consider Fig. 2, which depicts LIDAR returns from just two adjacent beams accumulated over several seconds of vehicle motion along a known trajectory. Here, we color one beam in red and the other in white; it is apparent that due to the LIDAR’s movement through space, to a large extent both beams end up hitting the very same surfaces.

---

<sup>1</sup> If neither the extrinsic nor intrinsic calibration is known precisely, then the two separate calibration procedures can be performed iteratively until both converge.

The location of the LIDAR unit relative to the vehicle's coordinate frame will be expressed with an x (longitudinal), y (lateral), and z (height) offset along with roll, pitch, and yaw angles. The (0, 0, 0) reference point and reference orientation is specified by the coordinate system being used, i.e. the three-dimension point and orientation that the vehicle's positioning system considers to be the origin.

In contrast to existing methods, our approach makes no assumptions about the environment other than that it is generally static and contains some 3D features, i.e. is not just smooth ground. In order to achieve an accurate calibration, we record LIDAR measurements as the vehicle transitions through a series of known poses.<sup>2</sup> Global pose information is irrelevant, as there is no existing map, so only local pose information is required. Local pose data may be acquired in any number of ways, e.g. from a wheel encoder and IMU, from an integrated GPS/IMU system, or from a GPS system with real-time corrections. Again, it is only the relative motion of the vehicle through the trajectory that is relevant to the calibration, so no global pose data is necessary.

Now, we define an energy function on point clouds which penalizes points that are far away from surfaces defined by points from other beams:

$$J = \sum_{b_i=1}^B \sum_{b_j=b_i-N}^{b_i+N} \sum_k w_k \|\eta_k \cdot (p_k - m_k)\|^2 \quad (1)$$

where  $B$  is the total number of beams and  $N$  is the number of neighboring beams we align each beam to,  $k$  iterates over the points seen by beam  $b_j$ ,  $p_k$  is the  $k$ th point projected according to the current transform,  $m_k$  is the closest point to  $p_k$  seen by beam  $b_i$ ,  $\eta_k$  is the surface normal at point  $m_k$  and  $w_k$  is 1 or 0 based on whether  $\|p_k - m_k\| < d_{max}$

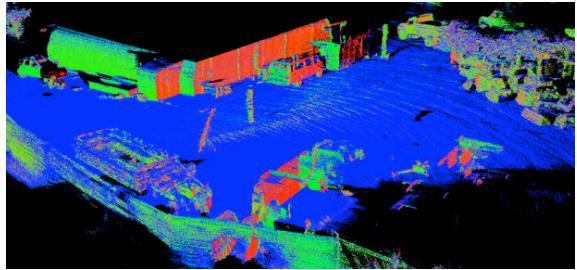
This energy function bears similarity to the point-to-plane iterated closest point (ICP) error function [7], with two key differences. First, we compare surfaces defined by points in each beam individually against points in neighboring beams. This has the crucial benefit that an erroneous calibration between beams will not significantly affect surface normals in the set of points seen by any individual beam. Second, unlike in ICP, we are not dealing with rigid point clouds, as a change in any calibration parameter will transform the points in that beam's point cloud in a complex way, since in this case the points in each cloud were observed at different times and thus from different poses of the sensor.

Surface normals are computed separately per beam, by fitting a plane to each point's 20 nearest neighbors in the accumulated projected points from the entire trajectory. Due to the density of data from multi-beam LIDARs, this local neighborhood for each point is very small. We show an example of these surface normals

---

<sup>2</sup> The vehicle trajectory can be arbitrary, but must include a change in yaw so that lateral and longitudinal offsets of the LIDAR can be detected; if the vehicle only moves straight, these cannot be disambiguated. Similarly, we do not attempt to recover the sensor's height, as our vehicle always remains nearly parallel to the ground, though height can trivially be determined by considering the distance to points the vehicle drove over.

**Fig. 3** Accumulated points colored by computed surface normal; the red, green, and blue channels respectively are set to the surface's x, y, and z components of the normal vector at each point.



in Fig. 3, where the red, green, and blue channels are colored according to normal vector's x, y, and z components at each point.

A further benefit of the high density of points returned by multi-beam LIDARs is that almost any surface will be nearly locally planar at the resolution of the point cloud; thus, projecting points from one beam onto the surfaces defined by points in neighboring beams results in very low errors when all calibrations are accurate.

Given the above energy function, all that remains is to select the extrinsic calibration that minimizes the total score. Although in theory the objective is not necessarily convex, and thus finding the true global optimum cannot be guaranteed in any reasonable amount of time, in practice the energy function is quite smooth and standard search heuristics perform very well.

In our approach, we alternatively optimize the translation parameters and rotation parameters until both have converged. For each optimization, we utilize grid search, which compares the current energy score with the score that results from adjusting the variables in question in all possible directions jointly. Whereas a coordinate descent iteration takes time linearly proportional to the number of variables, grid search takes time exponential in the number of variables, as it considers all combinations of directions. On the other hand, it is less prone to getting stuck in local minima, and as neither translation nor rotation individually has more than three variables, grid search is computationally tractable. For example, when considering a rotation change, each of roll, pitch, and yaw can be increased, held constant, or decreased, which results in 26 new comparisons to the current score.<sup>3</sup>

We start with a relatively large step size, iterate until convergence, and repeatedly reduce the step size until we've reached the finest granularity we desire. At the end of the last optimization, we obtain our final calibration parameters.

---

<sup>3</sup> It is important to note that for every possible calibration considered, all points must be reprojected into 3D space based on the vehicle's pose at the time each point was acquired. Thus, unlike with ICP, a calibration change does not warp or distort all points in the same way, as the effect of a calibration change on each individual point depends on where the vehicle was at the time that return was measured. The projection of a laser return into a 3D point is computed simply by adding the return's distance, in the direction the beam points, to the laser's origin in 3D space; the laser's origin is located at the pose of the vehicle's reference frame at the time of measurement plus the extrinsic transform for the laser relative to the vehicle frame.

### 3 Intrinsic Calibration of Each Beam

The motivation in the previous section applies equally to the case of intrinsic calibration. That is, an intrinsic calibration that computes each beam’s horizontal and vertical angle and range offset correctly will necessarily yield a lower energy score than an incorrect calibration.

It is worth emphasizing that this property is a direct consequence of the fact that the vehicle moves during data collection. For a stationary vehicle, it is impossible to disambiguate certain calibrations; indeed, many possible angles and range offsets may be equally plausible in that case. But when the vehicle moves in a known trajectory, no longer will incorrect calibrations result in plausible point clouds when each beam’s returns are accumulated over time and projected appropriately into 3D space.

Although the energy function used to calibrate the sensor’s extrinsic pose is equally applicable to its intrinsic calibration, it is intractable to perform grid search over 3 parameters for each of tens or hundreds of beams jointly. Instead, we alternately consider all horizontal angles, all vertical angles, and all range offsets until convergence. At each step, for the variables in question, we take empirical derivatives of the energy function across pairs of beams with respect to the individual parameters. Consider again the energy function:

$$J = \sum_{b_i=1}^B \sum_{b_j=b_i-N}^{b_i+N} \sum_k w_k \|\eta_k \cdot (p_k - m_k)\|^2 \quad (2)$$

At each iteration, for each beam  $b_i$  and neighboring beam  $b_j$  we hold fixed the accumulated projected point clouds and accompanying surface normals associated with beam  $b_j$  and then re-project the points from beam  $b_i$  with the parameter in question increased and then decreased by some increment  $\alpha$ . For each of the two possibilities, the inner part of the energy function,  $\sum_k w_k \|\eta_k \cdot (p_k - m_k)\|^2$  is recomputed; the parameter is then changed by  $\alpha$  in whichever direction improves the objective maximally, or else the parameter is held constant if all perturbations are worse.

In this manner, we iteratively loop through all parameters and beams, optimizing the objective function at each step, until either some predetermined number of iterations is reached or until the change in the global objective function becomes sufficiently small. We note that although this heuristic works well in practice, unlike with grid search for extrinsic calibration, it is not actually guaranteed to lower the objective function in any given iteration, as it updates a particular parameter for all beams in each iteration. Given the extremely large search space, however, such approximations are reasonable, and, as we show in the results section, work very well in practice.

## 4 Remittance Calibration

In addition to estimating the LIDAR's pose and beam parameters, we also derive a Bayesian generative model of each beam's response to surfaces of varying reflectivity using Expectation Maximization. [8]

As the vehicle transitions through a series of poses, let  $T$  be the set of observations  $\{z_1, \dots, z_n\}$  where  $z_i$  is a four-tuple  $\langle b_i, r_i, a_i, c_i \rangle$  containing the beam ID, range measurement, intensity measurement, and map cell ID of the observation, respectively. The map may be comprised of 2D cells in which points are projected to the ground plane, or full 3D correspondences can be used. As we have shown in previous work [6], the deterministic calibrated output  $c(a, j)$  of beam  $j$  with observed intensity  $a$  can be computed in a single pass as follows:

$$c(j, a) := E_{z_i \in T} [a_i \mid ((\exists k : c_i = c_k, b_k = j, a_k = a), b_i \neq j)] \quad (3)$$

That is, the calibrated output when beam  $j$  observes intensity  $a$  is the conditional expectation of all other beams' intensity readings for map cells where beam  $j$  observed intensity  $a$ .

Taking this further, we can derive a probabilistic calibration that specifically models the uncertainty and noise characteristics of each beam, which in practice are often very different. We note that although environment reflectivity is of course continuous, for computational reasons we restrict the values to integers between 0 and 255, as this matches the range of the Velodyne's returns. Thus for each map cell  $c$  we maintain a discrete distribution  $P(m; c)$  indicating the probability that map cell  $c$  has intensity  $m$ , for  $m = 0$  to 255.

Now, for each beam  $b$  we wish to estimate a distribution  $P(a|m; b)$  indicating the probability that beam  $b$  will return intensity  $a$  given that its measurement comes from a map cell with intensity  $m$ . Each map cell is initialized with a uniform intensity prior, and for each beam we initialize the prior

$$P(a|m; b) = \eta \cdot [\exp\left(\frac{-(a-m)^2}{2\tau}\right) + \varepsilon] \quad (4)$$

where  $\eta$  is the normalizer,  $\tau$  controls the peakiness of the distribution, and  $\varepsilon$  affords a nonzero probability of a random intensity return. Note that while each beam maintains its own distribution  $P(a|m; b)$ , all beams are initialized with the same 256 x 256 distribution; with this initialization, *a priori* beams are likely to return values near the true brightness of the map.

Let us define  $\theta$  as the beam model, where  $\theta$  is the collection of  $P(a|m; b)$  for all 64 beams. Thus, in our implementation,  $\theta$  is a 64 x 256 x 256 table. Similarly, let us define  $M$  as the intensity distribution over map cells, where  $M$  is the collection of  $P(m; c)$  for all cells in the entire map. Thus, in our implementation,  $M$  is a 256 x  $|M|$  table, where  $|M|$  is the number of cells in our map. We note that after calibration,

$M$  can be discarded, as it is ultimately our beam model  $\theta$  that we are interested in computing.

Starting with the initializations above we alternate between computing  $P(m; c)$  for each map cell (E-step) and computing  $P(a|m; b)$  for each beam and map intensity (M-step). We note that while the intensities of each map cell are by no means independent of each other, because they are jointly affected by the beam models, if we make the approximation of ignoring spatial correlations between nearby map cells, then they become conditionally independent of each other given the beam models. Since we are ultimately interested in deriving the beam models rather than the map, and given the abundance of data, ignoring potential spatial correlations between map cells is feasible and makes the problem tractable. The update equations are as follows.

#### 4.0.1 E-Step

The E-step updates each map cell independently:

$$P(m_k = m|z; \theta) = \eta \cdot \prod_{i:c_i=k} P(a_i|m; b_i) \quad (5)$$

Thus, in the Expectation step we compute the distribution over intensities for each map cell given the sensor data and holding the current beam parameters constant.

#### 4.0.2 M-Step

The M-step updates the beam model for each beam independently. First, we compute:

$$C(m, a; M, b) = \sum_{k=1}^K P(m_k = m) \cdot \mathbf{1}\{\exists i : b_i = b, c_i = k, a_i = a\} \quad (6)$$

where  $C(m, a; M, b)$  is the expected count in the data of beam  $b$  observing intensity  $a$  for cells of map intensity  $m$ , given the distribution over map cells in  $M$  computed in the previous E-step. Then, using these expected counts, we compute the maximum a posteriori (MAP) estimate for the beam parameters:

$$P(a|m; M, b) = \eta \cdot C(m, a; M, b) \cdot P(a; b) \quad (7)$$

where  $P(a; b)$  is the prior probability of beam  $b$  observing intensity  $a$ . Using Bayes' rule, this equation computes the distribution over possible intensity return values for each beam given the distribution over map cell intensities. Thus, in the Maximization step we obtain the most likely beam parameters given the sensor data and holding the current distribution over the intensities for each map cell constant.

After EM converges, we obtain a fully generative model for each beam's response to environment surfaces of different reflectivities.

## 5 Experimental Results

We demonstrate the performance of the calibration algorithms presented here with several experiments. We used a Velodyne HD-64E S2 LIDAR sensor with 64 beams oriented between -22 to +2 degrees vertically and rotating 360 degrees horizontally. The unit spins at 10Hz and provides around 100,000 points per spin. This sensor was mounted to the roof of our research vehicle as shown in Fig. 1 (left).

In addition, our vehicle pose was provided by an Applanix LV-420 positioning system that combines GPS, IMU, and wheel encoder readings to provide global and inertial pose updates at 200 Hz. However, as the methods described here only require locally consistent pose data, we ignored the global GPS pose values and only used the unit's local 6-DOF velocity updates, which we integrated over time to produce a smooth local coordinate frame.

We implemented our algorithms in C, taking advantage of the University of Maryland's Approximate Nearest Neighbor (ANN) library [5]. For the following results, we used a maximum matching distance of 20cm and generated per-beam point clouds and surface normals using all laser returns, but only evaluated the energy function at every 16 points for efficiency; with over a million points returned by the Velodyne per second, it is unnecessary to evaluate the energy function at every single point. With this implementation on a modern desktop processor, using about 15 seconds of recorded data, the extrinsic and intrinsic calibrations each take on the order of one hour to converge, given a very bad initialization. The remittance intensity calibration is faster, requiring a few minutes to run.

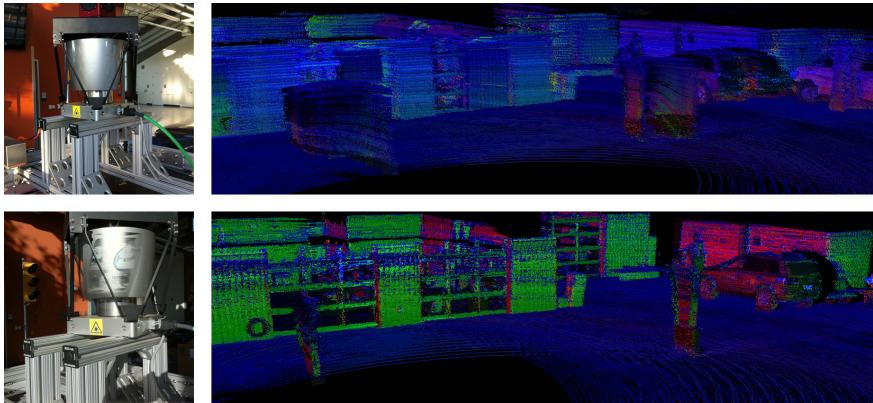
### 5.1 *Extrinsic Calibration*

First, we show that we can precisely and reliably compute the Velodyne's mounting location on our vehicle, even with a poor initialization. As discussed previously, we attempt to recover the sensor's lateral and longitudinal offset and roll, pitch, and yaw relative the the vehicle's coordinate frame.

We collected two short 15-second logfiles with each of two different mounting positions for the Velodyne, for a total of four logfiles. In all four cases, we drove the vehicle in a tight semi-circular arc close to a building at .2 m/s.

First, we ran the calibration routine on the two logfiles taken with the initial mounting position. Here, the Velodyne sensor was centered laterally on the vehicle, and positioned 1.51 meters forward of the rear axle, which is our positioning system's reference origin. It was pointed straight forward and mounted parallel to the roof rack, as shown in Fig. 4.

On the first logfile, starting with an initial calibration that was within 10cm and 1° of the measured location, our algorithm computed a lateral position of 0.00m,



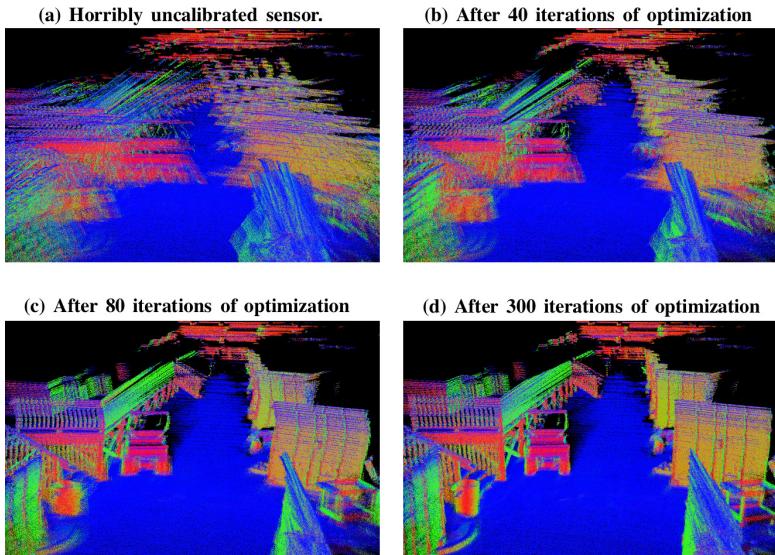
**Fig. 4** Recalibrating a moved sensor. The Velodyne was moved from its original mounting position (top left) to a new position, including translation and rotation (bottom left). Using the old calibration for the new position, projected Velodyne points show massively deformed structures and significant blurring of surfaces (top right). After calibration, the points are much better aligned (bottom right), and the computed transform for the Velodyne is extremely accurate.

a longitudinal position of 1.51m, a roll of  $-.03^\circ$ , a yaw of  $0.00^\circ$ , and a pitch of  $-.46^\circ$ . On the second logfile, with the Velodyne in the same location, the algorithm computed a lateral position of -0.01m, a longitudinal position of 1.50m, a roll of  $-.03^\circ$ , a yaw of  $0.03^\circ$ , and a pitch of  $-.46^\circ$ . Thus, from two separate drives, from two different locations, the resulting position estimates were within 1 cm and  $.03^\circ$  of each other in all dimensions.<sup>4</sup>

For a more challenging test, we then remounted the Velodyne 5.6 cm to the right and 20.6 cm behind the original location, and we rotated it counter-clockwise (around the Z axis) by  $9$  to  $10^\circ$ , as shown in Fig. 4. Now, starting with the calibrated pose from the original mounting location, we ran our algorithm on each of the two new logfiles. On the second of the new logfiles, it correctly estimated that the sensor had been moved by 6 cm to the right and 21 cm to the rear, and rotated by  $9.78^\circ$  counter-clockwise. The dramatic improvement in the resulting 3D point cloud, comparing the assumed original mounting location to the estimated new location, is shown in Fig. 4.

On the first of the new logfiles the estimate was less accurate; the computed offsets were 16cm to the right and 30cm to the rear, along with  $9.56^\circ$  counter-clockwise. Thus, although the directions of the movement were correct, the amounts were not. Upon examining this logfile, we discovered that there had in fact been

<sup>4</sup> We are only able to measure the mounting angles to within  $1^\circ$ , so we cannot empirically verify the angle calibration to within  $.03^\circ$  granularity to which we estimate angles in our algorithm; however, both the energy functional and the surfaces in the resulting 3D point clouds are optimal with the computed calibration and degrade noticeably if they are changed in either direction.



**Fig. 5** Unsupervised horizontal angle and range calibration using 10 seconds of data (all scans depicted above). Points colored by surface normal. Even starting with an unrealistically inaccurate calibration (a) we are still able to achieve a very accurate calibration after optimization (d).

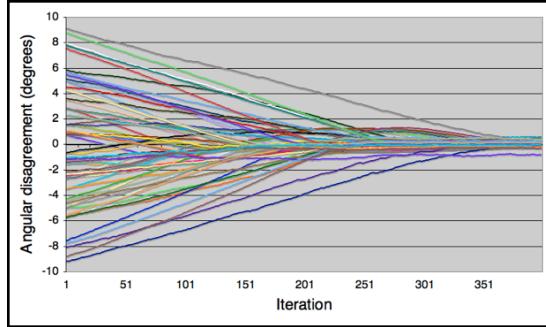
IMU drift across the trajectory, which resulted in the accumulated 3D point cloud showing visible smearing. Indeed, the inaccurate Velodyne pose estimate our algorithm computed not only had a better scoring energy function than the correct calibration for that logfile, but it also visually resulted in less smearing than the correct calibration. In this case, our algorithm picked the extrinsic calibration that resulted in the "best" 3D point cloud, but because the assumption of a correct local trajectory was violated, the estimated calibration was a bit off.

Therefore, these results demonstrate that our algorithm produces extremely accurate extrinsic pose estimates when the trajectory is known precisely, and that its performance degrades when the trajectory estimate is inaccurate.

## 5.2 *Intrinsic Calibration*

Next, we show that we can accurately calibrate the Velodyne sensor's individual beam angles and range offsets. Recent improvements in Velodyne's meticulous supervised factory calibration give better results than earlier models; not only can our algorithm do better still, we show that we are able to take an artificially bad calibration and use our methods to arrive at a calibration whose accuracy exceeds the best available factory calibration. Fig. 5 depicts such an optimization; starting with an unrealistically bad calibration in which we set all horizontal angles to be 0 and all

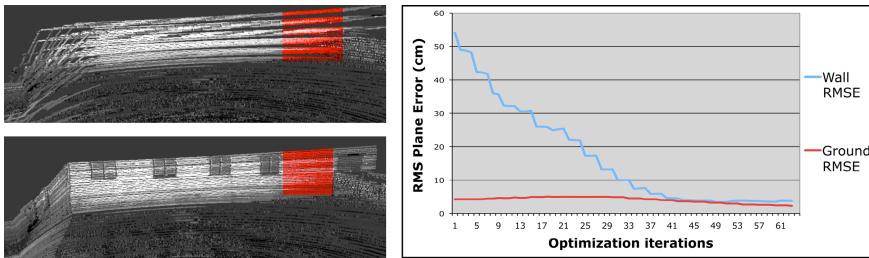
**Fig. 6** Comparing the horizontal angles of all 64 beams with the factory calibration, starting with a uniform initial estimate and optimizing over 400 iterations. Initially beams were miscalibrated by up to  $9^\circ$ , and after calibration all beams' angles agreed with the factory calibration to within  $1^\circ$ , with an RMS deviation of only  $0.25^\circ$ .



range offsets to be equal,<sup>5</sup> we are able to recover an excellent calibration based on only 10 seconds of data in a complex unlabeled environment.

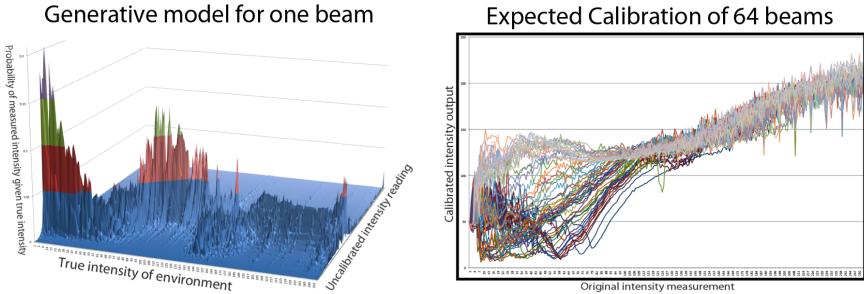
We show quantitative success in two ways. First, in optimizing the intrinsic calibration, the angles and range offsets we compute are very similar to the factory values, even when initialized to be significantly different. In one experiment, we optimize the horizontal angles starting from a uniform initial estimate of  $0^\circ$  per beam. With this initialization, the beam angles disagree with the factory calibration by up to  $9^\circ$ , with an RMS disagreement of  $4.6^\circ$ ; after our optimization, the maximum disagreement is less than  $1^\circ$ , with an RMS disagreement of  $0.25^\circ$  (Fig. 6). Thus our result is, for all beams, very similar to the factory calibration. Indeed, although we are unable to measure angles to these tolerances, we find the resulting energy functional and visual appearance of our calibration both outperform the factory calibration, suggesting that much of the disagreement may be due to factory miscalibration, or slight angle shifting over time.

Going further, in Fig. 7 we see another application of our calibration in a simpler environment, after which we hand-selected an area of horizontal ground and



**Fig. 7** Wall planarity improves from bad initialization (top left) after calibration (bottom left). Graph depicts quantitative improvement in wall planarity during calibration procedure (right). Final result exceeds factory calibration for both wall and ground.

<sup>5</sup> In reality, the horizontal beam angles range from  $-9^\circ$  to  $+9^\circ$  within the Velodyne and the range offsets range from 0.85m to 1.53m.



**Fig. 8** Learned intensity models. (Left) shows a generative model  $P(a_i|m)$  for beam 37 of 64. (Right) shows the expected environment intensity given each beam’s intensity return. All 64 beams are shown here; note significant variation between beams.

vertical wall known to be planar. We compare the RMS distance of the points to their approximating plane over the course of optimization. With our artificially bad calibration we start with ground and wall errors of 54 and 4 cm, respectively, and after optimization these are reduced to 4 and 2 cm, respectively. Importantly, the baseline Velodyne factory calibration gives errors of 6cm and 3cm, respectively, so our unsupervised method provides superior results.

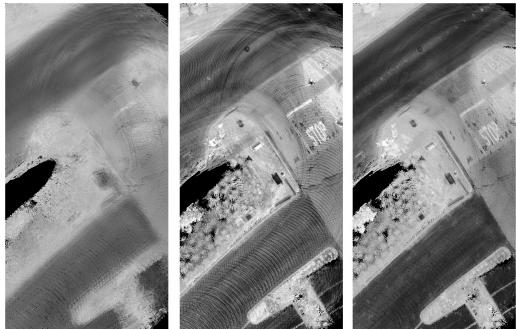
### 5.3 Bayesian Remittance Calibrations

The results of Bayesian intensity calibration can be seen in Fig. 8 (left), in which we plot the learned generative model for one of the 64 beams. Here we see, as expected, that brighter surfaces tend to yield brighter returns. The somewhat surprising non-monotonicity of the graph corresponding to the brightest 45% of surfaces may be partially explained by the fact that fewer than 0.1% of the Velodyne returns fall into that brightness region. Thus, there is very little data to generate that section of the graph; at the same time, this phenomenon causes that region of the response function to rarely be queried.

Each beam has a significantly different intensity response; we show the expected intensity values of the environment given each beam’s measured intensity in Fig. 8 (right).

Finally, we show the significant impact of angle, range, and intensity calibration on the resulting laser maps. In Fig. 9 we show an orthographic intensity map of points projected to the ground plane, first with our artificially bad calibration, then with calibrated angles and ranges, and lastly adding calibrated intensities. The final calibrated map displays excellent sharpness and contrast, indicative of a well-calibrated sensor.

**Fig. 9** Improvement from calibration. We compare an orthographic intensity map of a street with the horrible angle and range calibration used in Fig. 5 (left), the same map with the learned angles and ranges (center), and finally adding intensity calibration (right). The final result is much improved.

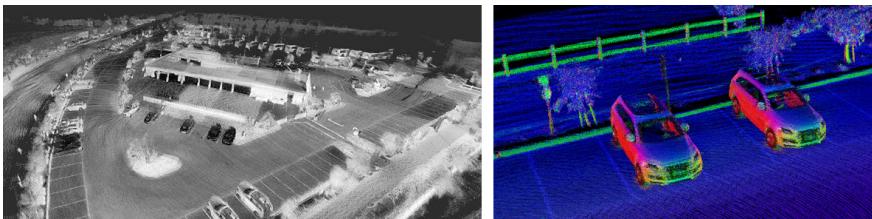


## 6 Conclusion

Multi-beam LIDAR sensors are a key enabling component of advanced mapping and robotic applications, and their use will only increase with time. We have presented what we believe to be the first complete fully unsupervised calibration algorithms for these sensors, and have demonstrated that excellent results are achievable with a trivial amount of data collection in arbitrary unlabeled environments even with terrible initializations. In addition, these methods are extensible and have many related applications. For example, these algorithms could easily be extended to the calibration of multiple single-beam LIDARs mounted to one vehicle platform.

Although the algorithms we discuss here make particularly few assumptions about the environment, they do treat the environment as static, i.e. we assume it is only the data collection vehicle that is moving. To the extent that there are dynamic obstacles during data collection, this motion could interfere with the results; in these cases, existing segmentation and tracking algorithms could be employed [12, 13] to remove such tracks, although an especially poor initial calibration may render segmentation and tracking more difficult than usual.

In this paper we presented a solution to a particular instance of the Simultaneous Calibration and Mapping (SCAM) problem, in which neither a calibration nor a map is available *a priori*. Such situations are in fact common in practice, despite being less studied than the more popular Simultaneous Localization and Mapping (SLAM) problems. This discrepancy is perhaps due in part to the fact that reasonable



**Fig. 10** Mapping a parking lot with SLAM and SCAM, using 60 seconds of accumulated data. Points colored by intensity return (left), and closeup of the scene with points colored by both intensity and surface normal (right).

calibrations for simple sensors are often obtainable by hand measurement, whereas SLAM problems cannot be solved similarly. In addition, SCAM - or at least the solution presented here - is inapplicable to single-beam sensors as they alone do not provide enough data for fully unsupervised calibration in the general setting.

A natural extension would be to combine SCAM with SLAM; that is, to solve Calibration, Localization, and Mapping jointly when none are known precisely. In preliminary work, we have aligned a several-minute logfile with both SLAM and SCAM as presented here; a resulting 3D point cloud can be seen in Fig. 10. However, a general solution to jointly optimizing all unknowns remains elusive, particularly if the pose estimates are of poor quality.

## References

1. Chao, G., Spletzer, J.: On-Line Calibration of Multiple LIDARs on a Mobile Vehicle Platform. In: ICRA 2010 (2010)
2. Censi, A., Marchionni, L., Oriolo, G.: Simultaneous maximum-likelihood calibration of odometry and sensor parameters. In: ICRA 2008 (2008)
3. Kaboli, A., Bowling, M., Musilek, P.: Bayesian calibration for Monte Carlo localization. In: AAAI 2006 (2006)
4. Muhammad, N., Lacroix, S.: Calibration of a rotating multi-beam Lidar (2009), <http://www.pges.fr/2rt3d/SortedDocs/Publications/CalibrationOfAMultiBeamLidar2.pdf>
5. Mount, D., Arya, S.: ANN: A Library for Approximate Nearest Neighbor Searching, <http://www.cs.umd.edu/~mount/ANN/>
6. Levinson, J., Thrun, S.: Robust Vehicle Localization in Urban Environments Using Probabilistic Maps. In: ICRA 2010 (2010)
7. Chen, Y., Medioni, G.: Object Modeling by Registration of Multiple Range Images. In: Proc. of the 1992 IEEE Intl. Conf. on Robotics and Automation, pp. 2724–2729 (1991)
8. Dempster, A., Laird, P., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society. Series B 39(1), 1–38
9. Segal, A., Haehnel, D., Thrun, S.: Generalized-ICP. In: Robotics Science and Systems (2009)
10. Underwood, J., Hill, A., Scheding, S.: Calibration of range sensor pose on mobile platforms. In: Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA (October 2007)
11. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press (2005)
12. Lee, K., Kalyan, B., Wijesoma, S., Adams, M., Hover, F., Patrikalakis, N.: Tracking random finite objects using 3D-LIDAR in marine environments. In: Proceedings of the 2010 ACM Symposium on Applied Computing (2010)
13. Shackleton, J., VanVoorst, B., Hesch, J.: Tracking People with a 360-degree Lidar. In: 7th IEEE Conference on Advanced Video and Signal Based Surveillance (2010)
14. Petrovskaya, A., Thrun, S.: Model based vehicle detection and tracking for autonomous urban driving. Autonomous Robots 26(2-3) (April 2009)
15. Douillard, B., Brooks, A., Ramos, F.: A 3D Laser and Vision Based Classifier. In: International Conference in Intelligent Sensors, Sensor Networks and Information Professing, ISSNIP 2009 (2009)
16. Steinhauser, D., Ruepp, O., Burschka, D.: Motion segmentation and scene classification from 3D LIDAR data. In: Intelligent Vehicles Symposium. IEEE (2008)