MSG-Cal: Multi-sensor Graph-based Calibration

Jason L. Owens* — Army Research Laboratory
Philip R. Osteen* — Engility Corporation
Kostas Daniilidis — University of Pennsylvania

***Abstract*— We present a system for determining a global solution for the relative poses between multiple sensors with different modalities and varying fields of view. The final calibration result produces a tree of transforms rooted at a single sensor that allows the fusion of the sensor streams into a shared coordinate frame. The method differs from other approaches by handling any number of sensors with only minimal constraints on their fields of view, producing a global solution that is better than any pairwise solution, and by simplifying the data collection process through automatic data association.**

I. INTRODUCTION

Most researchers who work with robots are familiar with the requisite yet error-prone process of determining the poses of multiple sensors in order to fuse the sensor data into a single reference frame. Therefore, a valuable goal would be the creation of a system in which a single data collection can automatically produce a globally optimized calibration of an arbitrary sensor configuration. In this work we describe a method and system that computes the relative poses for multiple environment sensors with differing modalities and varied acquisition rates using graph optimization.

A significant aim for sensor fusion research in robotics is to obtain a calibration procedure that can be run interactively and online using only the capabilities of a mobile robot and its current environment. While an algorithm that does not require a calibration object would be ideal, the data association problem makes this difficult to achieve, which is made even more challenging by differing sensing modalities. Therefore, we constrain the data association problem by making use of a custom calibration object, shown in Figure 3a, with the expectation that using the object will facilitate accurate calibrations.

The system we present makes few assumptions about the number, relative pose or fields of view of the sensors; in fact, the only requirement is that any single sensor has a partially overlapping field of view with at least one other sensor, so that the calibration object is always observed by at least one pair of sensors at the same time. The sensor type determines the representation of the object that is detected and stored at each distinct pose of the target. We assume that the calibration target is planar, which allows us to construct the geometric relationships between individual sensor detections required for calibration. Finally, we assume that each sensor has been intrinsically calibrated, and do not include intrinsic parameters as part of the problem formulation.

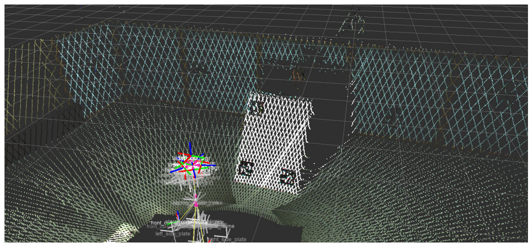


Fig. 1: The top image shows an example colored 3-D LRF frame from the calibration data collection.

The calibration procedure can be divided into three phases: data collection, pairwise calibration, and graph-based global refinement. In short, the data collection phase involves walking around the robot with the calibration object, and using a remote trigger to indicate a frame capture. The frame is then transformed into a set of edges where each edge represents a sufficient number of co-occurring detections between a pair of sensors. Given the edges collected during the previous phase, the pairwise calibration phase computes the relative pose between sensor pairs using RANSAC to filter outliers. Finally, the inliers from the previous phase are used to construct a hypergraph that uses non-linear optimization to generate a global pose graph solution.

Through both simulated quantitative results and real-world qualitative results, we show that the resulting framework successfully calibrates a number of different types of sensors with minimal operator intervention, and the global optimization is shown to be more accurate than the initial pairwise calibration.

A. Related work

The fundamental challenge in sensor calibration is determining associations between each sensor's data. Recent calibration approaches use either known scene geometry (including specialized calibration targets), or attempt to calibrate in arbitrary environments. Systems that calibrate with no special scene geometry or calibration object, such as [1] and [2], require high quality inertial navigation systems (INS) to compute trajectory.

*These authors contributed equally to this work.
Corresponding author: jason.l.owens.civ@mail.mil

Scaramuzza et al. are able to calibrate a 3-D laser and a camera without a calibration object or inertial sensors, by having a human explicitly associate data points from each sensor [3]. In contrast, our calibration framework uses only the data coming from the sensors we wish to calibrate, and does not require human assisted data association.

Similar to the work of [4], [5], [6], [7], [8] and others, we also use a calibration object to facilitate automatic data association. To our knowledge, there has been little work in the calibration of an arbitrary number of various types of sensors. Most work on calibration has been limited to either a single pair of different sensor types ([4], [8], [5], [1], [3], [9], [7], [6]), or multiple instances of a single type ([10], [2]). An exception is the work of Le and Ng [11], who also present a framework for calibrating a system of sensors using graph optimization, though their graph structure differs fundamentally from ours; they require that each sensor in the graph be a 3-D sensor. Therefore, neither individual cameras nor 2-D laser scanners are candidates in their calibration. Instead they must be coupled with another sensor (e.g., coupling two cameras into a stereo pair) that will allow the new virtual sensor to directly measure 3-D information. Their results agree with ours, that using a graph formulation reduces global error when compared with incrementally calibrating pairs of sensors. Our approach is related to the graph optimization proposed in [12] where the focus is on solving non-linear least squares on a manifold. Our approach provides initial estimates from pairwise solutions, and it has been tested on a graph of five sensors with three different modalities.

B. Paper contributions

We describe a flexible framework for accurately calibrating and fusing data from an arbitrary configuration of camera, laser, and point cloud sensors, where initialization estimates are generated automatically from the collected data. In addition, our system for collecting data requires only one operator, who needs no domain-specific expertise and need not tediously associate data points. We describe a unique formulation of the calibration problem as a hypergraph containing both sensors and observations as separate vertices, incorporating geometric constraints between the different detection modalities. Finally, we developed a simulation system that produces all three types of data used for the calibration and can be used to produce benchmark data sets for future calibration research.

II. PROBLEM DESCRIPTION

We address the extrinsic calibration of multiple sensors with various modalities and acquisition timescales that are rigidly mounted to a robotic vehicle.

In this context, calibration means the estimation of relative sensor poses such that features detected by multiple sensors can be fused into a single coordinate frame.

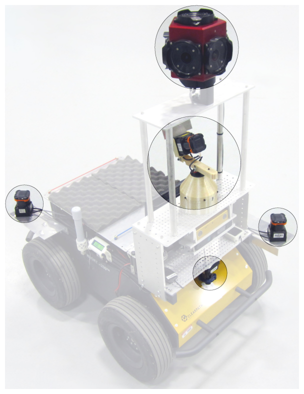


Fig. 2: The robot sensors we calibrate in this paper. The image highlights the three 2-D Hokuyo LRFs, the spinning Hokuyo 3-D LRF, and the Ladybug5 spherical camera.

Due to sensor noise and systematic feature estimation uncertainty, it's unlikely that features will align without error, so the process must be constructed to mimimize the error across all sensors.

In our case, we consider visible light cameras, 2-D planar laser range finders (LRFs) such as the Hokuyo^{TM}UTM-30, and 3-D laser range scanners (such as the Velodyne^{TM}HDL-32), or moving 2-D LRFs such as the one shown in Figure 2. We work with two geometric objects over the three sensor types: 3-D lines derived from 2-D LRFs and 3-D planes derived from both the cameras and 3-D LRFs. The calibration target itself is a large planar poster (as seen in Figure 3a) with fiducial tags for the cameras. Section III describes the object detection methodology for each sensor type.

The rest of the paper is organized as follows: the subsequent three sections (III-V) describe each phase of the procedure in detail. We briefly describe the simulation we used to produce quantitative results in section VI. We then discuss the experimental results in section VII and finally conclude in section VIII.

III. DATA COLLECTION

Data collection proceeds by launching a capture process with access to the robot's sensor streams. The user physically places the calibration object in various positions around the robot to capture sufficient views across all the sensors. Unfortunately, the sensors we are calibrating do not all support time synchronization. Therefore, to effectively bypass the problem of temporal data association, we require the user to hold or place the board in a fixed position until all sensors have taken at least two measurements. The requirement for two measurements is for sensors with long integration times (several seconds), where one cannot be assured the board was not in motion during the capture of the first reading. This implies that the time the calibration object must remain still is at most twice the period of the slowest integrating sensor, which for most sensors will be less than one second. Thus, there is no need to specify the data rates, or provide synchronization between the sensor acquisition processes.

We use the term *message* to refer to a single logical datum from a sensor, i.e. an image from a camera, a scan from a 2-D LRF, or a point cloud from a 3-D LRF. The term *frame* is used to refer to a collection of messages from every sensor. When the calibration object is in place for a new frame, the user sends a trigger to the system and data collection begins.

Each frame collected contains *all* the sensor data. While it may be possible that every sensor observes the calibration object, more likely only two or more sensors see it at one time. We must detect and extract the calibration object geometry for each sensor in order to correctly produce the edges in the sensor graph. At this stage, the system has no knowledge of the relative sensor positions, nor of the location of the calibration object in the environment, so given an image, point cloud or laser scan, the system must automatically detect the object for each sensor if it is in view.

A. Background subtraction

Since we rely on explicit correspondences between calibration object detections across sensors, we would like to have some assurance that the object we detect for each sensor in a frame is in fact the calibration object. We solve this problem using a background subtraction approach, which practically eliminates the data association problem and avoids constraints related to sensor modality.

At the very beginning of data collection, we allow the point cloud and line sensors to build up a model of the background environment in order to *subtract* it from the live dynamic data generated by the calibration target. This technique is typically used in imaging systems for surveillance, object tracking, and background replacement [13]. In our algorithm, we use an analogous technique for 3-D point clouds provided by PCL [14] that takes advantage of a modified octree. This octree is a spatial data structure that hierarchically subdivides volumes into octants and also provides efficient change detection. We accumulate points from the 3-D sensors over a short period of time, and store the observations in the octree. When data collection begins we create a new octree for each message from the sensors, and then subtract the accumulated background, leaving only the calibration operator and target (and possibly a few other noisy points) in the scene. Due to our usage of AprilTag fiducials on the calibration object, we do not need any explicit technique for background subtraction for cameras, since the tag detection automatically takes care of that.

B. Line and plane extraction from point clouds

Given a sparse point cloud from the background subtraction process, we use the following process to find the calibration object plane. First, sparse points are removed using a statistical filter, normals are computed for each point using the local neighborhood and then clusters are discovered by growing regions seeded by a point with the minimum curvature. For each cluster, the largest plane is extracted using random sampling consensus. The plane that best matches the known dimensions of the calibration object is selected as the valid detection for this sensor. In practice, there are few clusters due to the background extraction (usually related to the user's body or other stray points in the environment that may not have been captured by the background accumulation), and all but one of these clusters is too small to be the object.

Line extraction from the 2-D laser range finders proceeds in much the same way as the plane extraction, however, instead of using a plane model, we use a line model with RANSAC. For each detected line, found using loose thresholds to enable high recall, we ensure it is contiguous and within the required size bounds. Finally, we filter the egregious outliers, then refit a line to the remaining points. The endpoints we use to define the detected line segment are determined by projecting the actual scan endpoints to the closest point on the model line. The resulting line segment is reported as the detection for this sensor.

C. Plane extraction from AprilTag

As described in section II, the calibration target is primarily a planar object used to induce lines and planes from 2-D and 3-D LRFs, but we need to also be able to detect the plane from a camera. In the general case, cameras are projective devices, and can only determine an object pose up to scale. The AprilTag algorithm utilizes the known size of the tag, planar homography, and the camera parameters to determine the 3-D pose of each tag. We utilize three tags for redundancy, and use as many tags as are visible to determine the plane parameters. If more than one tag is visible, we currently compute the average normal and distance to origin.

IV. PAIRWISE CALIBRATION

The pairwise calibration phase estimates the $SE(3)$ transforms between each pair of sensors with co-occuring detections, using RANSAC to filter outliers. Since we convert all detections into the geometric primitives of either planes or lines, then in order to estimate the transform between each pair, we must solve one of the following objective functions.

A. Plane to plane

We use the Hessian normal form of a plane $P_i = \{\hat{\mathbf{n}}_i, d_i\}$ where $\hat{\mathbf{n}}_i^T\mathbf{x} = -d_i$. Therefore, if $\mathbf{x}$ is a point on plane P_i, then $\hat{\mathbf{n}}_i^T\mathbf{x} + d_i = 0$. As in [15], we note that the rotation and translation are separable problems. The normals are related by the rotation ${}^iR_j$ alone: $\hat{\mathbf{n}}_i = {}^iR_j\hat{\mathbf{n}}_j$. Using the plane equation, we can define the relations:

$$\begin{aligned}
\hat{\mathbf{n}}_i^T({}^iR_j\mathbf{x}_j + {}^i\mathbf{t}_j) &= -d_i \\
\hat{\mathbf{n}}_i^T\,{}^iR_j(-d_j\hat{\mathbf{n}}_j) + \hat{\mathbf{n}}_i^T\,{}^i\mathbf{t}_j &= -d_i \\
-d_j\hat{\mathbf{n}}_i^T({}^iR_j\hat{\mathbf{n}}_j) + \hat{\mathbf{n}}_i^T\,{}^i\mathbf{t}_j &= -d_i \qquad (1)\\
\hat{\mathbf{n}}_i^T\,{}^i\mathbf{t}_j + d_i - d_j &= 0, \qquad (2)
\end{aligned}$$

where, in Equation 1 we utilize the fact the $\hat{\mathbf{n}}_i^T\,{}^iR_j\hat{\mathbf{n}}_j \approx 1$, since the corresponding normals are unit vectors and ${}^iR_j$ brings $\hat{\mathbf{n}}_j$ into the frame of $\hat{\mathbf{n}}_i$.

We use these constraints in the following objective function:

$$\min_{({}^iR_j,{}^i\mathbf{t}_j)\in SE(3)} \sum_{(P_i,P_j)\in\mathcal{C}} \left\|\hat{\mathbf{n}}_i - {}^iR_j\hat{\mathbf{n}}_j\right\| + \sum_{(P_i,P_j)\in\mathcal{C}} \left[\hat{\mathbf{n}}_i^T\,{}^i\mathbf{t}_j + d_i - d_j\right]^2, \qquad (3)$$

where $\mathcal{C}$ is the set of plane correspondences for a sensor pair. In our implementation, we use the Kabsch algorithm [16] for estimating the rotation between the normal sets. To determine the translation, we compute the least squares result of Equation 2. In this paper, we treat all the normals with equal weight. However in the future we plan to compute the uncertainty of the detected planes and use a variant of Wahba's algorithm to compute the optimal rotation as per the method discussed in [17].

B. Plane to line

In contrast to our previous work [15], we utilize an objective function that minimizes the distance of the segment endpoints to the corresponding plane:

$$\min_{(R,t)\in SE(3)} \sum_{(P_i,\ell_i)\in\mathcal{C}} \sum_{\mathbf{x}_i^j\in\ell_i} \left[\hat{\mathbf{n}}_i^T\mathbf{x}_i^j + d_i\right]^2 \qquad (4)$$

While we provide no initial estimate, we have discovered that the line to plane RANSAC algorithm performs best with a sample size of five in order to sufficiently constrain the degrees of freedom.

V. GLOBAL CALIBRATION

We now describe the graph optimization that achieves the global calibration. In the previous phase, we compute a pairwise relative pose for every pair of sensors that meet or exceed the minimum number of observations. In this phase, we construct a hypergraph composed of several node and edge types that exploit the pairwise relative transforms as an initialization for the global sensor pose graph. The distinction is subtle but important: in the robot frame, we must pick one sensor as the root of the transform hierarchy (most transform libraries, e.g. `tf` in ROS, require a transform tree), and then connect the other sensors to this root, as per the pairwise connections in the previous phase, effectively forming a spanning tree over the graph of sensors. In addition, the pairwise calibration phase uses no additional information (i.e. other co-occuring detections) in order to minimize the observation error over more than one path in the graph. The goal of the global graph approach is to incorporate all the information into a single optimization structure, courtesy of g2o [18]. In our formulation, the sensor poses are the unknowns we wish to estimate simultaneously in a global frame. We let the user pick one sensor that will act as the root of the graph (e.g. a sensor with the largest FOV) and therefore become the origin of the global frame. In order to estimate the initial poses, we construct a minimum spanning tree $T_{\mathbb{S}}$ based on the edge weight between the sensors, defined as the sum of the squared errors computed during the pairwise calibration.

In this work, our hypergraph G is defined as a set of nodes V and hyperedges E. We include three node types in $V = \mathbb{S}\cup\mathbb{L}\cup\mathbb{P}$: sensor ($\mathbb{S}$), line-observation ($\mathbb{L}$), and plane-observation ($\mathbb{P}$). The sensor node set $\mathbb{S}\subset SE(3)$ contains elements $\mathbf{x}^{\mathbb{S}}\in\mathbb{S}$, the unknown sensor poses, initialized as per the sensor-sensor transform computed from the spanning tree $T_{\mathbb{S}}$. The line-observations and plane-observations correspond to every inlier detection found during the pairwise calibration phase. Line-observations $\mathbb{L}\subset\mathbb{R}^3\times\mathbb{R}^3$ are elements $\mathbf{x}^{\mathbb{L}}\in\mathbb{L}$, the endpoints of the detected lines. Plane-observations $\mathbb{P}\subset\mathbb{R}^4$ are elements $\mathbf{x}^{\mathbb{P}}\in\mathbb{P}$, i.e. the normal and distance to the origin. Note that $\mathbb{L}$ and $\mathbb{P}$ nodes exist relative to their observing sensors, while the sensors are the only entities in the global frame (where the root of $T_{\mathbb{S}}$ is taken as the origin).

The objective of our graph optimization is to find the sensor pose set $\mathbb{S}$ that minimizes the error over all edges in the graph. One can think of the nodes $\mathbf{x}_i^\alpha\in V$, $\alpha\in\{\mathbb{S},\mathbb{P},\mathbb{L}\}$ as providing the data values where i is the node identity and α is the node type. The hyperedges $e_\gamma\in E$ provide relations on the data, where $\gamma\subset\{i : \mathbf{x}_i^\alpha\in V\}$. The edges are the observations from the sensors and we construct functions F_{e_γ} that represent the noisy constraints in the system. Since the graph contains three node types there are necessarily six corresponding binary edge types, however, we currently only make use of five since we have not yet implemented a line-to-line pairwise calibration procedure.

The functions F_{e_γ} compute the errors we wish to minimize, representing the degree to which the parameters $\mathbf{x}_i^\alpha,\ldots,\mathbf{x}_j^\beta$ satisfy the initial constraint μ_γ (computed as the edge type-specific sensor observation). $\Omega_{\alpha,\beta}$ represents the information matrix of the constraint.

$$\min_{\mathbb{S}} \sum_{e_\gamma\in G} F_{e_\gamma}(\mathbf{x}_i^\alpha,\ldots,\mathbf{x}_j^\beta,\mu_\gamma)^\top\Omega_{\alpha,\beta}F_{e_\gamma}(\mathbf{x}_i^\alpha,\ldots,\mathbf{x}_j^\beta,\mu_\gamma) \qquad (5)$$

The graph optimization is implemented as a sparse non-linear least-squares minimization over the error values produced by the graph edges as shown in Eq. 5. During optimization, the algorithm computes the Jacobians of the error functions, and takes small linear steps in the tangent space of the sensor node manifold, in order to distribute the error over the nodes as defined by their uncertainty. Each node type provides an appropriate step operator in the tangent space, and uses the exponential map as necessary to compute the corresponding point on the manifold.

In the following sub-sections, we define each error function F_{e_γ} given the edge type defined by the set of observation types in each node.

Sensor-sensor edge. The sensor-sensor edge represents the relative transform between the sensor nodes it relates. Our error is defined as the difference of the relative transform from the mean. We define the mean to be the initial relative transform we compute from the pairwise calibration phase for this pair of sensors. If $\mu_{ij} \in SE(3)$ is the initial relative transform between sensor pair $\mathbf{x}_i^{\mathbb{S}}$, $\mathbf{x}_j^{\mathbb{S}}$, then the edge error in the tangent space is defined as:

$$\log_{SE(3)}(\mu_{ij}^{-1}((\mathbf{x}_i^{\mathbb{S}})^{-1}\mathbf{x}_j^{\mathbb{S}})). \tag{6}$$

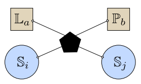
Sensor-line edge. The sensor-line edge represents the measured line endpoints relative to the sensor's coordinate frame. This edge constrains the adjustment of the line in the sensor frame with respect to the original observation. The edge uncertainty is related to the noise inherent in the laser scan, i.e. we assume the points (r, θ) detected by the 2-D LRF are perturbed by two independent normally distributed variables a, b as in:

$$p_i = (r + a, \theta + b) \tag{7}$$
$$a \sim \mathcal{N}(0, \sigma_r) \tag{8}$$
$$b \sim \mathcal{N}(0, \sigma_\theta). \tag{9}$$

The error is defined in a six dimensional space ($\mathbb{R}^3 \times \mathbb{R}^3$), with extremely low uncertainty in the z coordinate for each point. If μ_{ij} is the initially observed endpoints of the line j from sensor i, then the error is $\mathbf{x}_j^{\mathbb{L}} - \mu_{ij}$.

Sensor-plane edge. The sensor-plane edge is defined similarly to the sensor-line edge, but is parameterized by the four dimensional space ($\mathbb{R}^3 \times \mathbb{R}$) representing the plane parameters. Given μ_{ij} as the initially observed plane parameters, the error is $\mathbf{x}_j^{\mathbb{P}} - \mu_{ij}$.

Observation-observation edges. The following two hyperedges correspond to the unique constraints formed by the interaction between the geometric primitives. Each hyperedge is structured in the following manner:



where the squares are the observation nodes (line,plane) and the circles are the sensor nodes. The black pentagon represents the hyperedge. In every case, the observations are in the local frame of the corresponding sensor, which requires that we transform one of the objects into the frame of the other sensor. This is achievable since the sensor nodes represent their pose estimate in the global frame. We can compute the relative transform between the sensors and use this to bring the second observation into the frame of the first.

Line-plane edge. The line-plane edge connects a line and plane as detected by the respective sensor. In this case, we use the plane definition to represent the error in $\mathbb{R}^2$ as the offset of each of the line endpoints to the plane.

Plane-plane edge. Finally, the plane-plane edge connects the observations of the same plane from two different sensors. Given the transform of P_2 into the frame of P_1, we define the error in $\mathbb{R}^4$ as $(a_1, b_1, c_1, d_1) - (a_2, b_2, c_2, d_2)$, where a, b, c, d are the plane parameters.

VI. SIMULATION

In order to provide a benchmark quantitative evaluation of the proposed calibration procedure, we developed a simulation that generates the same raw sensor messages as the robot yet allows us to specify the ground truth poses for the sensors. The simulation includes a geometric representation of the calibration object derived from the generating PostScript program and realistic implementations of the target sensors: a pinhole camera that produces images of the object, a 2-D laser range finder to generate point clouds from the (r, θ) scan, and a spinning 2-D laser range finder that produces 3-D point clouds. We briefly describe the data set generation process, then follow with the implementation details of the sensors.

A. Data set generation

Similar to the collection procedure described in Section III, we generate randomly distributed poses for the calibration object, assuming the sensor system is situated at the origin. The procedure allows one to specify the bearing θ, radius r, maximum rotation around an axis ϕ and the number of random calibration object rotation samples to generate at each pose (θ, r),

$$\begin{aligned}\mathbb{R}_{(\theta,r)} = &\{(\psi_r, \psi_p, \psi_y)^i \mid \psi_r \sim U(-\phi, \phi),\\ &\psi_p \sim U(-\phi, \phi), \psi_y \sim U(-\phi, \phi)\}.\end{aligned}$$

For each sample, we "expose" the transformed calibration object to each simulated sensor and record the observations in a bag file.

B. Pinhole camera

The camera is modeled by the width and height of the generated image and the camera matrix:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{10}$$

We make the assumption that the camera is intrinsically calibrated, and do not generate any distortion. The image generation process is as follows: (1) a high resolution image of the calibration object is generated (i.e. 3 pixels per mm) directly from the metric specification of the poster size and embedded AprilTags. (2) Boundary coordinates are extracted from the transformed points of

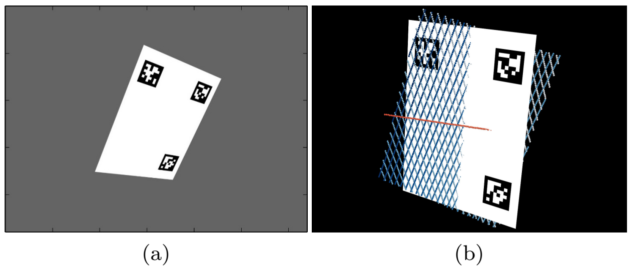


Fig. 3: (a) An image generated from a rotated calibration object at 2 m. (b) An example laser scan and point cloud rendered in 3-D before calibration, along with the ground truth object

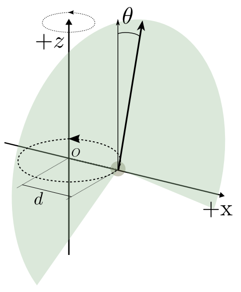


Fig. 4: The structure of the 3-D laser range finder. θ indicates the tilt (away from positive z), while d controls the offset from the axis of rotation, positive z. The shaded region indicates the vertical scan pattern from the 2-D LRF.

the calibration object, and are projected onto the image plane using the projection $\pi : \mathbb{R}^3 \to \mathbb{R}^2$:

$$\pi(\mathbf{x}) = \left(\frac{f_x x_1}{x_3} - c_x, \frac{f_y x_2}{x_3} - c_y \right). \tag{11}$$

Given the real image coordinates, we compute the homography matrix H that transforms the high-resolution model image to the planar deformation in the projected image. An example image is shown in Figure 3a.

C. 2-D laser range finder

We implement the 2-D LRF by intersecting rays cast from the origin of the sensor with the plane of the calibration object. Assuming the origin of the sensor is p_o, we construct a set of rays $\{\rho_i\}_1^N$ (normalized vectors) emanating from p_o with a configured angular resolution, start, and end angle. For each scan, we perturb the angles used to generate the rays, and once a valid intersection is found, we perturb the true radius according to Equation 9. We use the following ray intersection equation:

$$\hat{\mathbf{n}} \cdot (\rho_i t + \mathbf{p}_o) + d = 0 \tag{12}$$
$$\hat{\mathbf{n}} \cdot \rho_i t = -\hat{\mathbf{n}} \cdot \mathbf{p}_0 - d \tag{13}$$
$$t = \frac{-\hat{\mathbf{n}} \cdot \mathbf{p}_o - d}{\hat{\mathbf{n}} \cdot \rho_i}, \tag{14}$$

and discard intersections where $t < 0$, since that means intersection is behind the laser. With a value for t we can compute the point on the plane, and then determine whether it is within the calibration object bounds. This is accomplished by testing for negative intersections with all four half-planes defined by the boundary segment normals. These normals lie orthogonal to the plane normal and boundary segment, and emanate away from the centroid. Figure 3b shows an example laser scan in 3-D.

D. 3-D laser range finder

The 3-D LRF is implemented by "spinning" the 2-D LRF around a vertical axis and therefore generating multiple 2-D laser scans from different poses. We mimic the construction of the sensor shown in Figure 2, with the following parameters: tilt, offset, scans per degree, and scan angle. The first two parameters control the structure of the sensor, while last two parameters determine the density of the scans and the field of view (see Figure 4 for an illustration of the geometry). If one sets the offset and tilt both to zero, then the effect would be to simply rotate a 2-D LRF with the forward axis parallel to the z axis. Since the pose of the scan plane changes as the LRF rotates around the axis, we must transform each scan into the frame of the origin. Doing so generates the blue cross-hatched plane shown in Figure 3b.

VII. EXPERIMENTAL RESULTS

We evaluate the proposed system on two sets of sensors: one in which we know the ground truth, and can provide quantitative error metrics with respect to computed vs. actual position, and a set of sensors mounted on a mobile robot, in which we do not have ground truth. The calibration system is implemented as a set of C++ and Python packages based on the ROS ecosystem. The simulated and real datasets contain 120 and 133 different calibration object poses corresponding to one frame per pose, respectively.

The robot is a ClearPath Husky$^{\text{TM}}$with five primary sensors as shown in Figure 2. All three modalities are represented, including several 2-D LRFs, one slowly rotating 3-D LRF, and five cameras we use as one camera from the Ladybug5 spherical camera.

A. Error computation

We report our results using the following two metrics: error over the graph and deviation from ground truth (for the simulation study). The graph error is currently defined as the sum of the squared error over all edges other than the sensor-sensor edges (i.e. every sensor-observation and observation-observation edge). Thus, this error is related to *all* the data embedded in the graph. Every error is represented in terms of meters, so the error unit is m^2. For the simulation, we directly compare the sensor pose model with the resulting poses from the optimization procedures. We report these errors as the mean translational and angular offset (measured as the smallest angle between the principal axes of two sensors, i.e. the angle subtending the geodesic on the unit sphere S^2) from ground truth over all sensors.

We also investigated several variations on the algorithm to determine performance under different conditions. In particular, we were interested in the value of the pairwise calibration phase. First, we compared the global result when the sensors were initialized from the pairwise spanning tree transform vs. using the identity transform, and discovered this yielded no significant difference. However, in both cases the inlier set from the pairwise calibration was used. We wondered whether the results would be the same if we used all the data in the graph (no filter) vs. using the pairwise RANSAC inlier set (SAC filtered). The following two sections summarize our results for the simulation and real-world robot sensors.

B. Simulation

We use the simulation to show the algorithm works with respect to ground truth, given realistic simulated sensor data. Figures 5a and 5d show the benchmark error of the simulation configuration against the known ground truth sensor poses. One thing to note is that the variance of the graph solution is always lower than just the pairwise pose results over multiple runs. In addition, it is interesting to note that the mean of the angular error is lower than the graph (by 0.03 degrees), but this is not entirely unexpected given the error over the whole system: the pairwise solution represents the best relative pose given the data, but the global optimization has to correct for these errors across the entire graph. In Figure 5b, we see that the global graph error mean and variance are reduced as compared to the *best* pairwise solution with no global optimization. Finally, Figure 5c illustrates the performance over multiple runs while comparing the global optimization performance with and without filtering the outliers using the pairwise calibration phase. Note that while the filtered error shows small variance, the error is significantly lower when the outliers are filtered.

C. Robot Sensors

To test the algorithm using real-world sensors, we collected data from the robot and sensors shown in Figure 2. Since we do not have ground truth, we report the results of the graph error given the pose estimates from the pairwise calibration as compared to the error after the global graph optimization (Figure 5e). We also show the difference between filtering the outliers and using all the data with no filtering (Figure 5f). Note the agreement between the relative values as compared to the simulation results; graph optimization improves the error over the pairwise initialization, given the structure of the edge constraints. In addition, filtering the outliers significantly improves the error result.

However, we feel the qualitative results speak more clearly to the capability of the algorithm, and we show two screen captures from the 3-D visualization of the fused data. Figure 6a shows a large indoor area captured with the system with an inset of the corresponding image from the Ladybug's cameras. Figure 6b shows another scene from a highly cluttered machine shop in the same building, with small details that reinforce the practical performance of the system.

VIII. Conclusion

We have presented an end-to-end system for calibrating a set of minimally constrained multi-modal sensors rigidly mounted to a robot. Although many of the concepts have been presented in previous works, as far as we are aware this is the first algorithm to bring them together into a cohesive graph optimization framework paired with a simplified data association process. In the future, we plan to evaluate the algorithm on a wider array of data sets and explore improvements to the camera plane estimation algorithm. In addition, we will work to make the code available as an Open Source distribution for ROS so others can benefit and improve upon this work.

References

[1] J. Levinson and S. Thrun, "Automatic online calibration of cameras and lasers," in *Robotics: Science and Systems*, 2013.

[2] W. Maddern, A. Harrison, and P. Newman, "Lost in translation (and rotation): Rapid extrinsic calibration for 2d and 3d LIDARs," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 3096–3102, IEEE, 2012.

[3] D. Scaramuzza, A. Harati, and R. Siegwart, "Extrinsic self calibration of a camera and a 3d laser range finder from natural scenes," 2007.

[4] Q. Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, (Sendai, Japan), pp. 2301–2306, Oct. 2004.

[5] P. Núñez, P. Drews Jr, R. Rocha, and J. Dias, "Data fusion calibration for a 3d laser range finder and a camera using inertial data.," in *ECMR*, pp. 31–36, 2009.

[6] R. Unnikrishnan and M. Hebert, "Fast extrinsic calibration of a laser rangefinder to a camera," Tech. Rep. CMU-RI-TR-05-09, Robotics Institute, July 2005.

[7] H. Li and N. Fawzi, "Comprehensive extrinsic calibration of a camera and a 2d laser scanner for a ground vehicle," p. 24, July 2013.

[8] F. M. Mirzaei, D. G. Kottas, and S. I. Roumeliotis, "3d LIDAR-camera intrinsic and extrinsic calibration: Identifiability and analytical least-squares-based initialization.," *I. J. Robotic Res.*, vol. 31, no. 4, pp. 452–467, 2012.

[9] F. Vasconcelos, J. P. Barreto, and U. Nunes, "A minimal solution for the extrinsic calibration of a camera and a laser-rangefinder," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 11, pp. 2097–2107, 2012.

[10] S. Miller, A. Teichman, and S. Thrun, "Unsupervised extrinsic calibration of depth sensors in dynamic scenes," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[11] Q. V. Le and A. Y. Ng, "Joint calibration of multiple sensors," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 3651–3658, IEEE, 2009.

[12] R. Wagner, O. Birbach, and U. Frese, "Rapid development of manifold-based graph optimization systems for multi-sensor calibration and SLAM," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 3305–3312, IEEE, 2011.

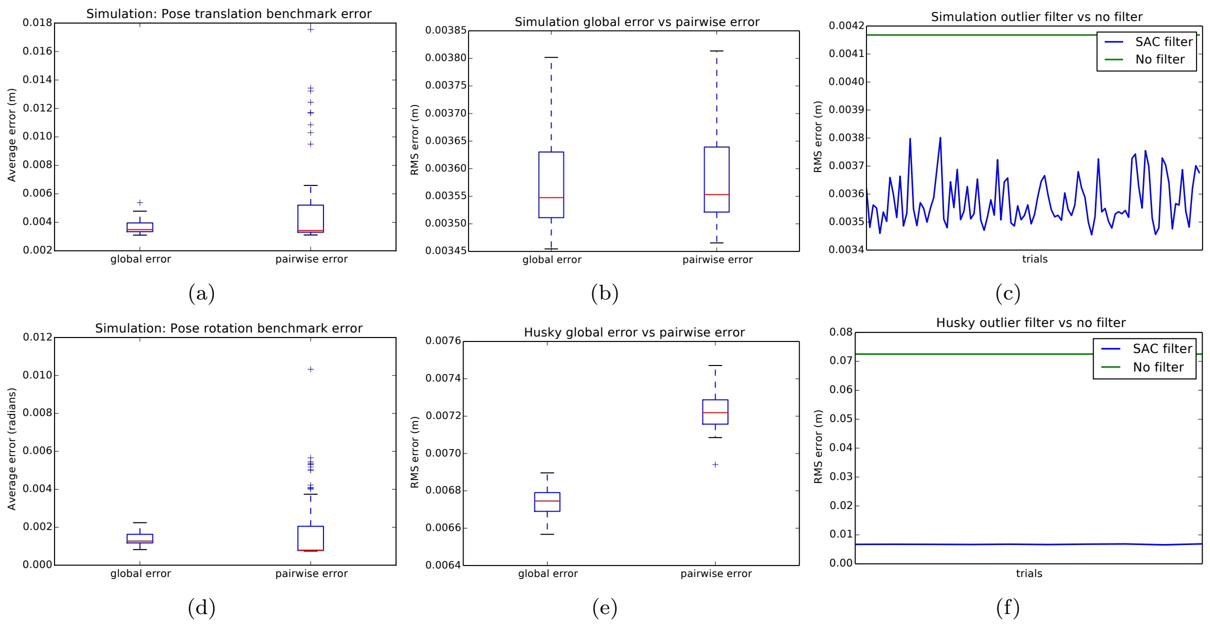
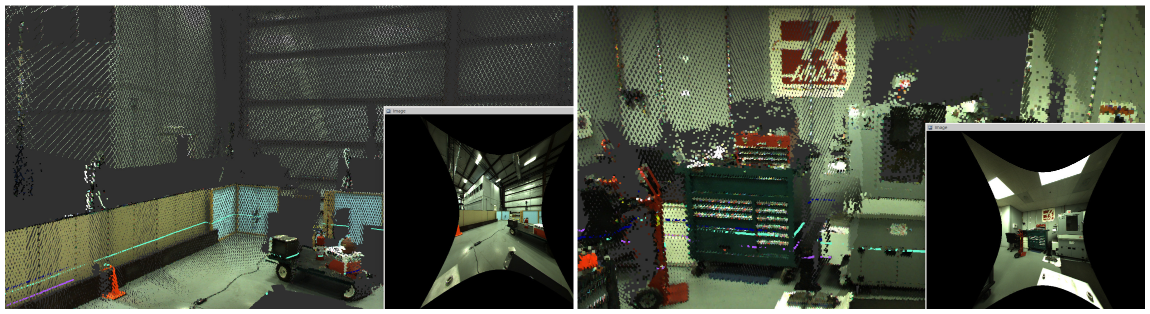


Fig. 5: (a) Simulation mean translation benchmark error (poses computed against ground truth). (b) Simulation graph error comparing the best pairwise error result vs. global error. (c) Simulation global graph error comparing results with and without filtering the data using the sampling consensus. (d) Simulation mean rotational benchmark error (poses computed against ground truth). (e) Husky graph error comparing the best pairwise error result vs. global error. (f) Husky global graph error comparing results with and without filtering the data using the sampling consensus.



(a) Indoor, open area scan. (b) Indoor, cluttered area scan.

Fig. 6: Fused sensor output using the global calibration results for the Husky sensor data collection. The Ladybug images are back-projected to the point cloud, given the relative transform between the two sensors. Also note the blue, purple, and cyan points that show the transformed 2-D laser scan output. In particular, (b) shows how well the laser scans align to the camera and 3-D LRF, e.g. by their intersection on the handcart on the left side of the image.

[13] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems*, vol. 25, pp. 1–5, 2001.

[14] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[15] R. Tron, P. Osteen, J. Owens, and K. Daniilidis, "Pose optimization for the registration of multiple heterogeneous views," in *Workshop on Multi-View Geometry in Robotics*, 2014.

[16] W. Kabsch, "A discussion of the solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, vol. 34, no. 5, pp. 827–828, 1978.

[17] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak, "Fast plane detection and polygonalization in noisy 3d range images," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 3378–3383, 2008.

[18] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3607–3613, IEEE, May 2011.