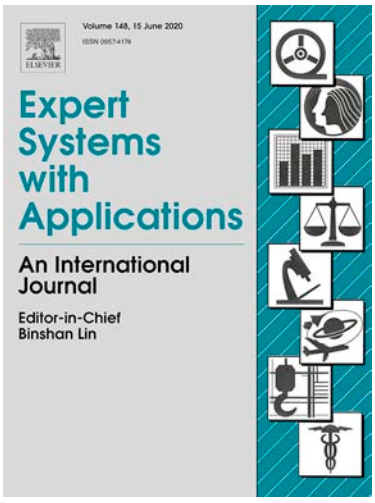


A Camera to LiDAR calibration approach through the Optimization of Atomic Transformations

André Silva Pinto de Aguiar, Miguel Armando Riem de Oliveira, Eurico Farinha Pedrosa, Filipe Baptista Neves dos Santos

PII: S0957-4174(21)00335-3
DOI: https://doi.org/10.1016/j.eswa.2021.114894
Reference: ESWA 114894

To appear in: *Expert Systems with Applications*

Received Date: 7 December 2020
Accepted Date: 6 March 2021

Please cite this article as: Pinto de Aguiar, A.S., Riem de Oliveira, M.A., Pedrosa, E.F., Neves dos Santos, F.B., A Camera to LiDAR calibration approach through the Optimization of Atomic Transformations, *Expert Systems with Applications* (2021), doi: https://doi.org/10.1016/j.eswa.2021.114894

A Camera to LiDAR calibration approach through the Optimization of Atomic Transformations

André Silva Pinto de Aguiar[a,c], Miguel Armando Riem de Oliveira[b], Eurico Farinha Pedrosa[b], Filipe Baptista Neves dos Santos[a]

[a] *INESC TEC - INESC Technology and Science; 4200-465, Porto, Portugal, {andre.s.aguiar, fbsantos}@inesctec.pt*
[b] *Institute of Electronics and Informatics Engineering of Aveiro, University of Aveiro, Portugal, {mriem, efp}@ua.pt*
[c] *School of Science and Technology, University of Trás-os-Montes e Alto Douro; 5000-801 Vila Real, Portugal*

Abstract

This paper proposes a camera-to-3D Light Detection And Ranging calibration framework through the optimization of atomic transformations. The system is able to simultaneously calibrate multiple cameras with Light Detection And Ranging sensors, solving the problem of Bundle. In comparison with the state-of-the-art, this work presents several novelties: the ability to simultaneously calibrate multiple cameras and LiDARs; the support for multiple sensor modalities; the calibration through the optimization of atomic transformations, without changing the topology of the input transformation tree; and the integration of the calibration framework within the Robot Operating System (ROS) framework. The software pipeline allows the user to interactively position the sensors for providing an initial estimate, to label and collect data, and visualize the calibration procedure. To test this framework, an agricultural robot with a stereo camera and a 3D Light Detection And Ranging sensor was used. Pairwise calibrations and a single calibration of the three sensors were tested and evaluated. Results show that the proposed approach produces accurate calibrations when compared to the state-of-the-art, and is robust to harsh conditions such as inaccurate initial guesses or small amount of data used in calibration. Experiments have shown that our optimization process can handle an angular error of approximately 20 degrees and a translation error of 0.5 meters, for each sensor. Moreover, the proposed approach is able to achieve state-of-the-art results even when calibrating the entire system simultaneously.

Keywords: Computer Vision, Geometric Optimization, Atomic Transformations

1. Introduction

Nowadays, autonomous robotic systems are endowed with high-quality onboard sensors of different modalities, i.e. sensors that output different types of data, such as cameras and Light Detection And Ranging (LiDAR) sensors. To move autonomously while safely avoiding any kind of obstacle, these vehicles need to perform complex tasks such as Simultaneous Localization and Mapping (Durrant-Whyte & Bailey, 2006) and Path Planning (Sariff & Buniyamin, 2006) which require calibrated sensor data. The quality of the onboard sensors data is also crucial, since the robot should have a clear perception of the environment. For example, in agriculture, the automation of tasks such as crop monitoring or harvesting is a complex challenge that requires data of high quality sensors (dos Santos et al., 2016), such as, for example long-range 3D LiDARs. To perform data fusion, and take advantage of all the sensors present in the robotic system, it is essential to know the spatial relationship between all sensors (Melendez-Pastor et al., 2017; Majumder & Pratihar, 2018). To do so, the most common approach is to perform *extrinsic calibration*, i.e., to compute the transformation between the sensors' reference frames. The standard approach to perform extrinsic calibration is to find associations between data incoming from each sensor to be calibrated. Thus, a cost function that minimizes the error between associations is used. Most of the calibration procedures use a pattern that can be detected independently of the sensor modality, so that data correspondences can be found. Using these concepts, camera to 3D LiDAR extrinsic calibration have been approached in several works. A minority of works perform calibration without using a pattern. In those, the characteristics of the environment are used as features to compute intrinsic and extrinsic calibration. In autonomous driving, for example, lane detection and vanishing point tracking are common approaches (Badue et al., 2021; de Paula et al., 2014; Álvarez et al., 2014).

The great majority of the works found in the literature follow a pairwise calibration between a monocular camera and a 3D LiDAR, a concept introduced by Huang & Barth (2009). In this work, the extrinsic coefficients are computed solving a closed-form equation, and refined with a maximum likelihood estimation. Similarly, Verma et al. (2019) use a standard chessboard to calibrate a perspective/fisheye camera and a 3D LiDAR using a Genetic Algorithm. Wang et al. (2017) propose a work where the corners of the pattern are automatically detected for both a panoramic camera and a 3D LiDAR so that the calibration can be performed. For the LiDAR case, authors propose a detection based on the intensity of reflectance of the beams. Fremont & Bonnifait (2008); Guindel et al. (2017a) use circle-based patterns to perform the extrinsic calibration. Mirzaei et al. (2012)

propose the estimation of a 3D LiDAR intrinsic parameters, as well as the extrinsic calibration with a monocular camera, through the minimization of a non-linear least squares cost function. The calibration is used to build photorealistic 3D reconstruction of indoor and outdoor scenes. Pandey et al. (2010) calibrate a 3D LiDAR with an omnidirectional camera also using a standard planar pattern. To calibrate the system, the sensors should observe the pattern from at least three different points of view. With this input, the extrinsic coefficients are calculated with a non-linear optimization technique. With the same purpose, Huang & Grizzle (2020) use a pattern of known dimension and geometry and estimates the pattern to LiDAR pose automatically using a fitting algorithm.

Although all these works perform successful extrinsic calibrations between 3D range sensors and monocular cameras, pairwise calibration is a major shortcoming since most robotic systems present more than two sensors to be calibrated. In a system with more than two sensors, one would have to combine multiple pairwise calibrations. The problem is that the number of the combinations of pairwise calibrations required grow quickly. For example, Zhou et al. (2018) presente a system with a 3D LiDAR and a stereo camera system. However, to calibrate the three sensors (LiDAR and two cameras), two calibrations have to be performed: LiDAR to left camera, and LiDAR to the right camera. In the same way, with the purpose of fusing point clouds of multiple stereo cameras, Dhall et al. (2017) use a 3D LiDAR to perform pairwise calibration with all the cameras in the system. Only after obtaining the transformation between the range sensor and each camera of the stereo system, the transformation between the stereo cameras can be found. Then the point clouds can be fused. Similarly, su Kim & Park (2019) perform six pairwise calibrations between a 3D LiDAR and six monocular cameras mounted in an hexagonal plate that constitute an omnidirectional camera.

To overcome this limitation, few works exist that consider multi-sensor, multi-modal calibration in a non-pairwise fashion. For example, Zuniga-Noel et al. (2019) propose a method to estimate the extrinsic calibration between multiple sensors such as LiDARs, depth cameras and RGB cameras. The calibration procedure is separated in two parts: a motion-based approach that estimates 2D extrinsic parameters and a method that uses the observation of the ground plane to estimate the remaining ones. It is worth noting that this framework requires the robotic platform to be moving. Liao et al. (2017) propose a joint objective function to simultaneously calibrate three RGB cameras with respect to an RGB-D camera. Rehder et al. (2016), propose an approach for joint estimation of both temporal offsets and spatial transformations between sensors. This approach is one of few

that is not designed for a particular set of sensors, since its methodology does not rely on unique properties of specific sensors. It is able to calibrate systems containing both cameras and LiDARs. Pradeep et al. (2014), present a joint calibration of the joint offsets and the sensors locations for a PR2 robot. This method takes sensor uncertainty into account and is modelled in a similar way to the bundle adjustment problem.

The two major shortcomings of the state-of-the art in extrinsic calibration are: most of the methods perform pairwise calibration, which can be exhaustive for a robotic system with many sensors to be calibrated; and the majority of the works are focused on specific sensor modalities, rather than working in a more general way. To overcome these issues, this work proposes an extrinsic calibration framework with the following contributions:

- Support for calibration of multiple sensors (i.e. $N \geq 2$);
- The ability to handle multiple sensor modalities;
- Calibration without changing the topology of the input transformation tree;
- Integration of the system within the Robot Operating System (ROS) framework, with interactive tools to collect data, set the initial estimates, and visualize the calibration.

Our previous works have focused on the calibration of intelligent vehicles. These platforms are often characterized by the large amount of sensors of different modalities mounted onboard. As such, these previous works presented a methodology based on atomic transformations for multi-sensor, multi-modal robotic systems (Guindel et al., 2017b; Oliveira et al., 2020a,b). In this work, we extend our framework - Atomic Transformation Optimization Method (ATOM) [1] (Oliveira et al., 2020a) - to also consider the calibration of 3D LiDARs along with the other supported modalities. Atomic transformations are geometric transformations that are not aggregated, i.e., they are indivisible. As such, this article presents the methodologies implemented to simultaneously calibrate a 3D LiDAR sensor with multiple cameras using a Bundle Adjustment optimization scheme (Agarwal et al., 2010). As our approach is not focused on a single robotic platform, we present the calibration of a different robotic platform in comparison with our previous works - the agricultural robot AgRob V16 (de Aguiar et al., 2020; Santos et al., 2019).

[1] `https://github.com/lardemua/atom`

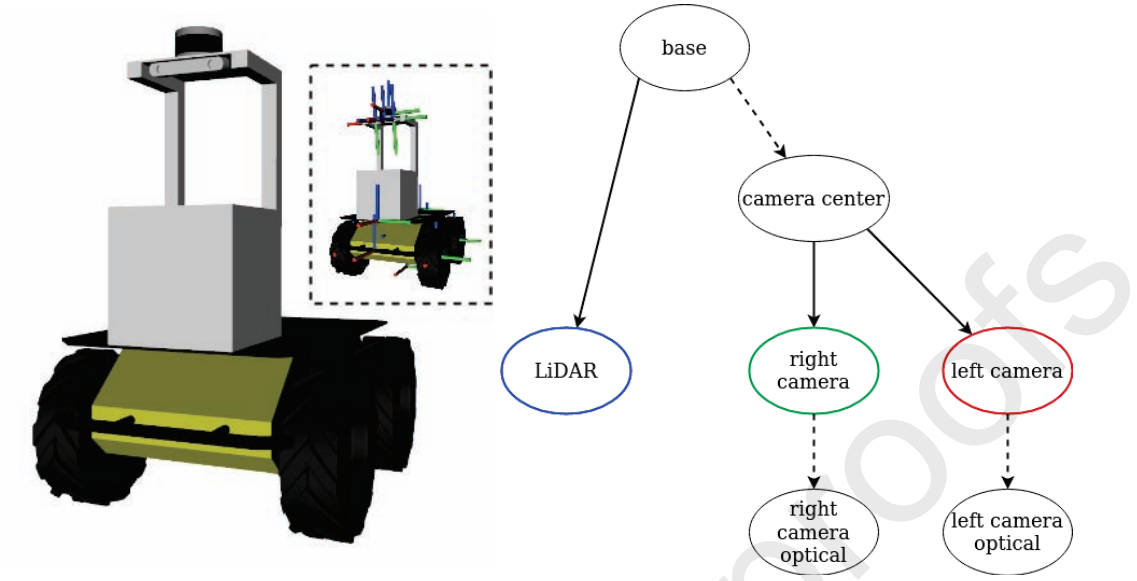
The remainder of this paper is organized as follows: Sec. 2 details the optimization procedure and how it is cast as a Bundle Adjustment problem; Sec. 3 describes the ROS (Quigley et al., 2009) calibration setup, i.e., the steps from the robotic platform configuration until the data collection; Sec. 4 details the experimental results; and finally, Sec. 5 provides conclusions and future work.

2. Proposed Approach

ATOM is a calibration framework that simultaneously calibrates sensors of different modalities though the optimization of atomic transformations. This concept is supported by a well-defined optimization pipeline, that defines a set of optimization parameters and minimizes a cost function that supports different input modalities. This function f, which depends on the optimization parameters $\mathbf{\Phi}$, is known as the *objective function*. Our approach minimizes f to calibrate all the sensors of generic multi-modal robotic platforms simultaneously. In this process, the definition of a tree graph which contains topological information about the relationship between reference frames is required. In this tree, nodes are reference frames and edges correspond to the transformation between nodes. This data structure allows for the definition of unique paths between the graph nodes, i.e., enables an efficient retrieval of the transformations between any two frames in the tree. Figure 1 represents the robotic platform to be calibrated (AgRob V16) with its respective referential frames, and the transformation tree considered for the calibration.

The design of the transformation tree leads to the definition of the optimization parameters to calibrate the system. An extrinsic calibration can be viewed as a pose estimation problem, where the pose of each sensor is estimated. Thus, the set of parameters to optimize $\mathbf{\Phi}$, must together define the pose of each sensor. To perform such a calibration, we propose to maintain the initial structure of the transformation tree, calibrating only one atomic transformation per sensor. Since the system contains camera sensors, it is also possible to introduce the intrinsic parameters of each camera in the set $\mathbf{\Phi}$. In this way, the set of parameters will be composed of different modalities and so, there is the need to design an objective function that is able to characterize sensors of in a multi-modal fashion.

As previously discussed, pairwise approaches for projecting the objective function result in complex graphs with many combinations of relationship definitions. For every existing pair of sensors, these relationships must be established according to the combined modality of the pair of sensors, which leads to a problem of scalability for which there is no solution in the literature. To solve



(a) (b)

Figure 1: **(a)** AgRob V16 model and the respective referential frames represented as red-green-blue axes. **(b)**: Transformation tree for AgRob V16 robotic platform. The majority of the frames not to be calibrated were omitted for simplicity. Each sensor has an associated atomic transformation, denoted by the solid edges. Dashed edges denote transformations that are not optimized (they can be static or dynamic). Each sensor has a corresponding link to which the data it collects is attached, denoted in the figure by solid thin ellipses. Very few approaches in the literature are capable of calibrating such a system while preserving the initial structure of the transformation graph.

this issue, we formulate our solution in Bundle Adjustment problem, in that the structure of the objective function is designed in *a sensor to calibration pattern* paradigm. Also, for every collection of data, the transformation that takes the corners of the calibration pattern to the world is optimized. In other words, the poses of the pattern are jointly estimated with the poses of the sensors. To perform this iterative procedure, the set of optimization parameters $\mathbf{\Phi}$ must be initialized. The first guess for the pattern pose is obtained w.r.t. one of the cameras to be calibrated, resulting in a transformation ${}^{cam_i}\boldsymbol{T}_p$, where p denotes pattern and ${}^{a}\boldsymbol{T}_b$ represents the transformation from frame b to a. As will be detailed later on, our calibration framework allows the definition of an initial guess for the pose of each sensor and, consequently, for the transformations to be calibrated. With this definition, it is possible to compute the pose of any particular sensor j as an aggregate homogeneous transformation ${}^{w}\boldsymbol{A}_{s_j}$, obtained from the chain of transformations for that particular

sensor present in the topological transformation graph:

$$ {}^{w}\boldsymbol{A}_{s_j} = \prod_{i \in R} {}^{i}\boldsymbol{T}_{i+1} = \prod_{i \in K} {}^{i}\boldsymbol{T}_{i+1} \cdot {}^{child}\hat{\boldsymbol{T}}_{parent} \cdot \prod_{i \in L} {}^{i}\boldsymbol{T}_{i+1}, \tag{1} $$

where ${}^{child}\boldsymbol{T}_{parent}$ represents the transformation to be calibrated, ${}^{i}T_{i+1}$ for $i \in K$ represent the prior links to the frame *parent*, ${}^{i}T_{i+1}$ for $i \in L$ the later to the frame *child*, and R is the set that contains all the frames present in the chain of transformation of sensor j. So, to obtain the homogeneous transformation from the pattern to the world, the following calculation is applied:

$$ {}^{w}\boldsymbol{T}_{p} = {}^{w}\boldsymbol{A}_{cam_m} \cdot {}^{cam_m}\boldsymbol{T}_{p} \ , \tag{2} $$

where p refers to the pattern, w states for world, and cam_m for the mth camera sensor. So, the set of parameters $\boldsymbol{\Phi}$ to be optimized is composed of the transformation represented in (2) along with the poses of each sensor to be calibrated, and, in the case of cameras, their intrinsics and distortion parameters:

$$ \boldsymbol{\Phi} = \Big[\overbrace{\boldsymbol{x}_{m=1}, \boldsymbol{r}_{m=1}, \boldsymbol{i}_{m=1}, \boldsymbol{d}_{m=1}, ..., \boldsymbol{x}_{m=M}, \boldsymbol{r}_{m=M}, \boldsymbol{i}_{m=M}, \boldsymbol{d}_{m=M}}^{\text{cameras}}, \\ \overbrace{\boldsymbol{x}_{n=1}, \boldsymbol{r}_{n=1}, ..., \boldsymbol{x}_{n=N}, \boldsymbol{r}_{n=N}}^{\text{LiDARs}}, \overbrace{\ldots}^{\text{Other modalities}}, \overbrace{\boldsymbol{x}_{k=1}, \boldsymbol{r}_{k=1}, ..., \boldsymbol{x}_{k=K}, \boldsymbol{r}_{k=K}}^{\text{Calibration pattern}} \Big], \tag{3} $$

where m refers to the mth camera to be calibrated, n states for the nth LiDAR to be calibrated, k refers to the pattern detection of the kth collection of data, $\boldsymbol{x}$ is a translation vector $[t_x, t_y, t_z]$, $\boldsymbol{r}$ is a rotation represented thought the angle-axis parameterization $[r_1, r_2, r_3]$ (where the vector $\boldsymbol{r}$ is used to represent the axis and its norm represents the angle), $\boldsymbol{i}$ is a vector with each camera intrinsic parameters $[c_x, c_y, f_x, f_y]$, and $\boldsymbol{d}$ is a vector of each camera distortion coefficients $[d_0, d_1, d_2, d_3, d_4]$. The intrinsic and distortion parameters of each camera can be initialized using any camera calibration toolbox, or in some cases, these parameters are also provided by the manufacture. The angle-axis parameterization was chosen because it has three components and three degrees of freedom, which means that it does not introduce more sensitivity than the one inherent to the problem itself (Hornegger & Tomasi, 1999), unlike the rotation matrix which has nine degrees of freedom for the also three components, or the euler angles that loose a degree of freedom when two axis are aligned. Using angle-axis representation, we have six optimization parameters per sensor that

represent the pose of each one, i.e., the geometric transformation that will be calibrated. The optimization procedure, as will be explained, consists of the minimization of an *objective function* by the definition of residuals that are calculated as an error (in pixels for RGB cameras and in meters for 3D LiDARs) between the re-projected position of the calibration pattern, and the position of the pattern detected by each sensor.

2.1. Objective function

To be able to consider multiple modalities in the same optimization process, we propose to structure the *objective function* $F(\mathbf{\Phi})$ as a composition of as many sub-functions $f_i(.)$ as desired modalities. The objective function $F(\mathbf{\Phi})$ from (4) is minimized using a non-linear least squares approach[2]. Least-squares finds a local minimum of a scalar cost function, with bounds on variables, by having an m-dimensional real residual function on n real variables. As such, we choose this minimization approach as its is the best fit for our problem. So, for each new modality added to the calibration, a sub-function associated with it is designed and incorporated in $F(\mathbf{\Phi})$, which allows for the minimization of the error associated with the pose of sensor of that specific modality. This is one of the reasons why the proposed approach is scalable. Thus, the optimization procedure can be defined as:

$$\underset{\mathbf{\Phi}}{arg\,min}\ F(\mathbf{\Phi}) = \underset{\mathbf{\Phi}}{arg\,min}\ \frac{1}{2}\sum_{i} \|f_i(\{\mathbf{\Phi_i}\})\|^2, \tag{4}$$

where $f_i(.)$ is the objective sub-function for the i-th sensor considering the set of k optimization parameters $\{\mathbf{\Phi_i}\}$. Thus, the final cost to be minimized is computed by the sum of the squared sub-function values for each set of optimization parameters. The value for all these sub-functions is a vector with the residuals associated to whit re-projection of the points of the calibrated pattern. For our use-case, the goal is to calibrate a stereo camera system (two cameras) and a 3D LiDAR sensor. So, the *objective function* is composed of the vector values of three sub-functions, two for the cameras and one for the 3D LiDAR. Each sub-function is detailed in the next sub-sections.

[2]In this work we used the least-squares solver provided by SciPy: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.least_squares.html`.

2.1.1. Camera modality sub-function

When the sensors to be calibrated are cameras, their calibration is performed as a bundle adjustment (Agarwal et al., 2010), as described in our previous work, Oliveira et al. (2020b). Thus, the created sub-function is based on the average geometric error corresponding to the image distance (in pixels) between a projected point and a detected one. So, the goal of the cost sub-function for camera sensors is to adjust the initial estimate for the intrinsic and distortion parameters, and position of the pattern corners, in order to minimize the re-projection error f_{cam}, given by:

$$f_{cam} = \Big[\|\boldsymbol{x}_{c=1} - \hat{\boldsymbol{x}}_{c=1}\|. \ . \ . \ , \|\boldsymbol{x}_{c=C} - \hat{\boldsymbol{x}}_{c=C}\|\Big], \tag{5}$$

where $\|.\|$ represents the Euclidean distance between two vectors, c is the index of the pattern corners, $\boldsymbol{x_c}$ denotes the ground-truth pixel coordinates of the measured points given by the pattern detection, and $\hat{\boldsymbol{x}}$ are the projected points, given by the relationship between a 3D point in the world and its projection onto the image.

To perform such calibration, 3D pattern points have to be found and re-projected onto the image plane. For each collection of data, the camera(s) to be calibrated capture the pattern. By knowing the real size of the pattern, and the size of each square that composes it, the 3D coordinates of the corners can be found in the local pattern reference frame. Then, each corner is located in the plane $z = 0$, since the corners are in the XoY plane of the local pattern reference frame. Thus, each corner in the local pattern referential frame $\boldsymbol{p}^p$ is transformed to the camera referential frame as follows:

$$\boldsymbol{p}^{cam} = {}^{cam}\boldsymbol{T}_w \cdot {}^{w}\boldsymbol{T}_p \cdot \boldsymbol{p}^p. \tag{6}$$

Note that, $\boldsymbol{p}^{cam}$ and $\boldsymbol{p}^p$ are homogeneous vectors of the 3D corner coordinates in each reference frame, so that (6) is valid. Finally, to re-project each 3D corner from camera's reference frame $\boldsymbol{p}^{cam}_{c=i}$ to the image plane, taking into account each camera intrinsic and distortion parameters, the pinhole camera model (Sturm, 2014) is used:

$$\hat{\boldsymbol{x}}_{c=i} = \boldsymbol{K} \cdot \boldsymbol{p}^{cam''}_{c=i}, \tag{7}$$

where $\boldsymbol{K}$ is the matrix that contains the intrinsic parameters $\boldsymbol{i}$ and,

$$\boldsymbol{p}_{c=i}^{cam'} = \boldsymbol{p}_{c=i}^{cam} \cdot \frac{1}{z_{cam}} = \left[\frac{x_{cam}}{z_{cam}},\ \frac{y_{cam}}{z_{cam}},\ 1\right]^T, \tag{8}$$

$$\boldsymbol{p}_{c=i}^{cam''} = \begin{bmatrix} x_{cam'} \cdot (1 + d_0 l^2 + d_1 l^2 + d_4 l^6) + 2d_2 \cdot x_{cam'} y_{cam'} + d_3 \cdot (l^2 + 2x_{cam'}^2) \\ y_{cam'} \cdot (1 + d_0 l^2 + d_1 l^2 + d_4 l^6) + d_2 \cdot (l^2 + 2y_{cam'}^2) + 2d_3 \cdot x_{cam'} y_{cam'} \end{bmatrix}, \tag{9}$$

where $l = \sqrt{(\frac{x_{cam}}{z_{cam}})^2 + (\frac{y_{cam}}{z_{cam}})^2}$, and d_j is the jth component of the distortion vector $\boldsymbol{d}$. From (6) to (9), it is possible to conclude that all the desired parameters to optimize are being considered: the pattern to world transformation ${}^{w}\boldsymbol{T}_p$ present in (6) that can be computed as the inverse of (2); the world to camera transformation ${}^{cam}\boldsymbol{T}_w$ also present in (6); and finally, the intrinsic $\boldsymbol{i}$ and distortion parameters $\boldsymbol{d}$ considered in (7)-(9).

The use of these parameters to project the 3D corners in the image plane, together with the minimization of the geometric re-projection error, lead to the parameter configuration that optimize the sub-function f_{cam}. Thus, it is expected that the re-projected points become closer to the ground-truth corners during the optimization. Figure 2 shows the difference between the initial position of the pattern corners, and the final position of these same projected points, after the optimization has been completed.

It is possible to observe that the pixels corresponding to the projection of the final position of the points (dots in Fig. 2) almost perfectly match the ground-truth point (squares in Fig. 2).

2.1.2. 3D LiDAR sub-function

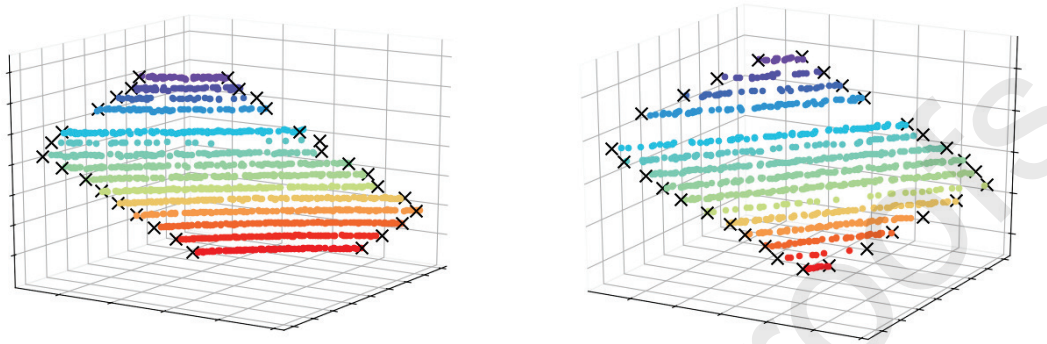
For the case of 3D LiDARs, the sub-function f_{lidar} considers two types of residuals: orthogonal distance and limit points distance. To compute both, this approach also uses the calibration pattern and, in specific, its boundary points. As will be detailed later on, our calibration framework has a semi-automatic labelling procedure that allows to save, for each collection of data, the LiDAR 3D points that are on the pattern. As in case of the camera sensor, this approach formulates the cost sub-function by minimizing the residuals w.r.t. some ground-truth. Here, the ground-truth 3D points are, once again, generated in the pattern reference frame by knowing the three dimensional structure of the pattern, such as its height and width. Thus, by knowing the size of the pattern, the size of each pattern square, and the pattern origin (bottom left corner), the coordinates of the boundary points defined in the local pattern's reference frame are computed. It is important to note that, the size of the board between the pattern grid and the end of the physical pattern had to

Figure 2: Difference between the initial position of the pattern corners, and the final position of these same projected points, after the optimization has been completed. Squares denote the position of the detected pattern corners; crosses denote the initial position of each projected corner; points denote the current position of the projected points.

be measured so that this step could be implemented. Also, as explained before, all the calculated pattern limit points have coordinate $z = 0$, since the pattern's reference frame is in the XoY plane. After calculating the ground-truth boundary points of the pattern, two things are required: the pattern boundary points observed by the 3D LiDAR sensor and the homogeneous transformation that converts 3D points from the LiDAR referential frame to the local pattern reference frame.

Given a set of labelled 3D LiDAR cartesian points on the pattern $\boldsymbol{p}^{lidar} = \left[x_{c=i}, y_{c=i}, z_{c=i}\right]$, the boundary points are calculated using a spherical parameterization for each 3D point. After computing the spherical coordinates of each 3D LiDAR point on pattern $\boldsymbol{p}^{s,lidar} = \left[r_{c=i}, \theta_{c=i}, \phi_{c=i}\right]$, two limit points are calculated considering the set of 3D LiDAR points on pattern belonging to a given horizontal scan of the original point cloud. As the labelled set of 3D LiDAR points on pattern is an unordered point cloud, the horizontal scans are computed by clustering the points considering their θ value. So, points with the same θ value belong to the same horizontal scan. Finally, to extract the two limit points per horizontal scan, the ϕ component maximum and minimum values

of each set are computed, resulting in the two most distant points, corresponding to points in the pattern boundaries. The result of this procedure is represented in Fig. 3.



(a) (b)

Figure 3: Two examples of the pattern boundary points extraction from LiDAR data. The colored points represent the clustered LiDAR scans considering the spherical component θ, and the crosses represent the boundary points extracted using the maximum and minimum values of the spherical component ϕ, for each cluster.

The final step before computing the residuals that constitute the cost sub-function f_{lidar} is to convert the set of labelled 3D LiDAR points on pattern, as well as the computed boundary points, to the patterns' reference frame. This is done using the homogeneous transformations computed in (1) and (2):

$$\boldsymbol{p}^{p} = {}^{p}\boldsymbol{T}_{w} \cdot {}^{w}\boldsymbol{A}_{lidar} \cdot \boldsymbol{p}^{lidar}, \tag{10}$$

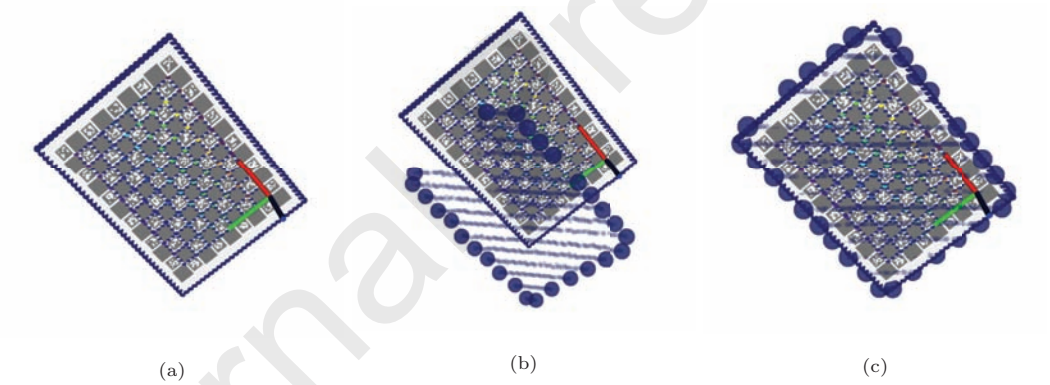
where ${}^{p}\boldsymbol{T}_{w}$ is the transformation matrix from the world to the pattern reference frame, and ${}^{w}\boldsymbol{A}_{lidar}$ the transformation matrix from the world to the LiDAR sensor reference frame. Similarly to the cameras' case, (10) shows that the optimization parameters include the sensor pose and the pattern pose. After this, the two residual types can be computed. The first, orthogonal distance, is the absolute z value of the coordinates of the projected 3D LiDAR points. As they are on patterns' referential frame, it is intended that their z coordinate is zero. Therefore, any value different from zero means that the optimization parameters (sensor pose and pattern pose) are not yet correct. The second residual type is the Euclidean distance of x and y components between the ground-truth pattern boundary points, and the LiDAR 3D points on pattern boundary calculated as described

before. For each LiDAR point on the pattern boundary, the residual is computed as the distance between x and y coordinates, in the calibration pattern frame, of the respective LiDAR boundary point and the closest point that belongs to the limit of the physical board that is being detected. This being said, the 3D LiDAR cost sub-function is as follows:

$$f_{lidar} = \left[\left\{ \left| z_{l=1}^{lidar,p} \right|, \| \boldsymbol{p}_{q=1,xy}^{boardlimit} - \boldsymbol{p}_{q=1,xy}^{p} \| \right\}, \ldots, \left\{ \left| z_{l=L}^{lidar,p} \right|, \| \boldsymbol{p}_{q=Q,xy}^{boardlimit} - \boldsymbol{p}_{q=Q,xy}^{p} \| \right\} \right], \quad (11)$$

where $z_{l=i}^{lidar,p}$ is the z coordinate of the ith 3D LiDAR point projected into the patterns' referential frame, $\boldsymbol{p}_{q=j,xy}^{boardlimit}$ are the x and y coordinates of the jth 3D LiDAR boundary point on the same referential, and $\boldsymbol{p}_{q=j,xy}^{p}$ are the x and y coordinates of the corresponding ground-truth boundary point.

Figure 4 shows the ground-truth pattern boundary points representation (blue lines on the left of Fig. 4), the calculated boundary points at the start of the calibration procedure (blue circles on the middle of Fig. 4), and the result of the optimization procedure with the ground-truth points and the boundary points observed by the LiDAR aligned (right representation on Fig. 4).



(a) (b) (c)

Figure 4: **(a)**: calibration pattern and respective ground truth boundary points represented as blue lines; **(b)**: misaligned boundary points observed by the 3D LiDAR with the ground-truth points at the start of the calibration procedure; **(c)**: optimization result - ground-truth points and projected boundary points aligned. It is noteworthy that the orthogonal distance aligned the z coordinate of the ground-truth pattern and 3D LiDAR points.

2.2. Normalization of multi-modal residuals

We propose a full calibration method where sensors of different modalities contribute to a global vector residual of residuals. While the camera sub-function provides a set of residuals that

are expressed in pixels, the LiDAR sub-function provides a set of residual expressed in meters. This mismatch in units of measurements may display highly disparities in error magnitudes, which could result in unwanted behaviours in the optimization processes due to differences in scale. For example, a residual of 1 pixel has higher influence (or weight) in the optimization path than a residual of 0.5 meters. Yet, our knowledge about the system tells us that the opposite should be considered. As result, the parameters that influence the residuals with higher scale will dominate the optimization, while the other parameters are perceived to already be close to their optimal state.

To handle the different scales in multi-model residuals in Equation 4, we employ a normalization factor to the optimization. Let $\mathcal{C} = \{\mathbf{c}\}$ be the set of existing residual classes (e.g $\mathcal{C} = \{\text{pixels}, \text{meters}\}$) and $c(i) \in \mathcal{C}$ is the residual class for the ith sensor, then the optimization is defined as

$$\underset{\mathbf{\Phi}}{arg\,min}\ F(\mathbf{\Phi}) = \underset{\mathbf{\Phi}}{arg\,min}\ \frac{1}{2}\sum_{i}\left\|\frac{f_i(\{\mathbf{\Phi_i}\})}{\eta_{c(i)}}\right\|^2, \tag{12}$$

where $\eta_{c(i)}$ is the normalization factor for residuals created by the sensor sub-function f_i. Note that the normalization factor $\eta_{c(i)}$ is defined per residual class and not per sensor. The normalization values per class are given by the arithmetic mean of the same class residuals before the optimization. For example, the normalization for the class *pixels*, with $\eta_{c(i)} = \eta_{\text{pixels}}$, is given by

$$\eta_{\text{pixels}} = \frac{1}{n}\sum_{j}\|f_j(\{\mathbf{\Phi_j}\})\|_1 : \forall f_j \in \{\text{pixels}\}\ , \tag{13}$$

where n is the total number of residuals that are part of the considered class and $\|.\|_1$ is the L1 norm. Note that the normalization factors are constant values during optimization, calculated once with the residuals that result from the initial guess.

3. Calibration framework

The ROS (Quigley et al., 2009) has become the standard framework for the development of robotic solutions. As referenced before, the proposed calibration procedure requires the creation of a transformation tree, from which atomic transformations are optimized. For this purpose, ROS provides a tree graph referred to as *tf tree* (Foote, 2013). With this tool, it is possible to define a data structure as the one present in Fig. 1. Also, the Robot Operating System Visualization (RVIZ) tool supports additional functionalities, such as robot visualization, collision detection, etc. In

fact, this visualization procedure is interactive, in that if any transformation between two links changes, the robotic platforms and sensors affected by these links change its pose accordingly. This interactive procedure is possible since the optimizations' cost function always recomputes the aggregate transformations. Therefore, a change in one atomic transformation in the chain affects the global sensor pose, and consequently, the error to minimize. So, if atomic transformations change due to the calibration procedure, the *tf tree* will automatically adjust the robots and sensors poses accordingly. It should be emphasized that, due to all these functionalities, the calibration procedure should not change the structure of the *tf tree*. Our approach preserves the predefined structure of the *tf tree*, since, during optimization, only the values of some atomic transformations contained in the chain are estimated, securing the topology of the tree. To the best of our knowledge, our approach is one of few which maintains the structure of the transformation graph before and after optimization.

Given all of the above, we state an extensive integration with ROS as a key component of the proposed approach. The ROS calibration framework is segmented in five main components: configuration, initial estimate, data labelling, data collection, optimization procedure. Each will be described in detail in the following sections.

3.1. Calibration configuration

The configuration defines the parameters which will be used throughout the calibration procedure, from the definition of the sensors to be calibrated to a description of the calibration pattern. The proposed approach, detailed in Sec. 2, is based on the optimization of atomic transformations. These were combined through the use of the topological information contained in a tree. The transformation tree is generated from a ROS Unified Robot Description Format (URDF) . Additional information must be given to define which, out of the set of atomic transformations, will be optimized during the calibration procedure. Also, a description of the calibration pattern must be provided. All this information is defined in a calibration configuration file.

3.2. Initial parameter estimation

Optimization procedures suffer from the known problem of local minima. This problem tends to occur when the initial solution is far from the optimal parameter configuration, and may lead to failure in finding adequate parameter values. To avoid this, the setup of the a plausible initial

guess for the entire parametric optimization system is essential. Our approach supports different modalities of parameters, as stated in (3). Thus, each modality requires a specific type of initialization. For cameras intrinsic $\boldsymbol{i}$ and distortion $\boldsymbol{d}$ parameters, the initialization is performed using any state-of-the-art camera calibration toolbox, or using the calibration provided by the manufacture. To initialize the transformations of sensors in general, we developed an interactive tool which parses the configuration URDF file and creates a 3D visualization tool for ROS interactive marker associated with each sensor. Figure 5 shows an example of the developed tool.

(a) (b)

Figure 5: Example of the developed interactive tool operation for AgRob V16 system. Here, the reference frames of the cameras that compose the stereo camera system and the 3D LiDAR sensor were moved. When the user positions the sensors in the desired pose, the initial estimate for the pose of each sensor is saved to use in calibration.

Here, we can see the user changing each sensor reference frame, dragging the respective interactive markers. With this tool the user can move and rotate the markers relative to each sensor. This provides a simple, interactive method to easily generate plausible first guesses for the poses of the sensors. Immediate visual feedback is provided to the user by the observation of the 3D models of the several components of the robot model and how they are put together, e.g. where each camera or LiDAR is positioned w.r.t. the vehicle. Also, for multi-sensor systems, it is possible to observe how well the data from a pair of sensors overlap. An example of this procedure can be watched at `https://youtu.be/llg8jYCeAjk`.

Concerning the atomic transformations associated with the calibration pattern ${}^{w}\boldsymbol{T}_{p}$ present in (2), these are initialized by defining a new branch in the transformation tree which connects the

pattern to the frame to which it is fixed. For example, for AgRob V16 case, ${}^{w}\boldsymbol{T}_{p}$ is estimated through (2) where ${}^{cam_m}\boldsymbol{T}_{p}$ is estimated solving the Perspectiva-n-Point (PnP) for the detected pattern corners (Gao et al., 2003; Fabbri et al., 2020; Penate-Sanchez et al., 2013), and ${}^{w}\boldsymbol{A}_{cam_m}$ by deriving its topology from the *tf tree* and using the initial values for each atomic transformation in the chain.

3.3. Labeling data

The labeling of data refers to the annotation of the portions of data which captures the calibration pattern. A labeling procedure is executed for the data of each sensor, and can be automatic, semi-automatic or even manual in some cases. The information that is stored depends on the modality of the sensor, but for cameras it is always the pixel coordinates of the corners observed in the pattern.

The standard calibration pattern that is used for camera calibration is a chessboard pattern. The images are labelled using one of the many available image-based chessboard detectors (Czyzewski, 2017). Our system is also compatible with charuco boards (Garrido-Jurado et al., 2016). These have the advantage of being able to detect the pattern even when it is partially occluded. Also in this case we make use of off the shelf detectors, e.g. Romero-Ramirez et al. (2018); Hu et al. (2019).

For the calibration of AgRob V16, a labeling algorithm for 3D LiDARs was developed. This method is semi-automatic and is initialized by a setup of a *seed point*. To label the pattern points viewed by the 3D LiDAR, the user drags an interactive marker to a point located in the pattern - the *seed point*. This constitutes the non-automatic stage of the procedure. After that, the algorithm clusters a set of points (that are intended to belong to the pattern) using an Euclidean distance threshold computed using the *a priori* known dimensions of the pattern. Despite being simple and fast, this approach reveals lack of precision, since it includes many outliers in the labeling procedure. The pattern used is rectangular. Thus, the Euclidean distance threshold has to be higher than the smaller side of the rectangle. This means that, if the pattern is close to another object, points from this object will be labeled as pattern points. To overcome this issue, a Random Sample Consensus (RANSAC) (Fischler & Bolles, 1981) algorithm is executed to fit the set of labeled points in a plane (the pattern plane), eliminating the outliers. RANSAC is an iterative algorithm, and it is performed a maximum number of times M. To find points that belong to the

plane, the point to plane distance is computed in each iteration i for each point j as

$$D_{ij} = \frac{\left| a_i x_j + b_i y_j + c_i z_j + d_i \right|}{\sqrt{a_i^2 + b_i^2 + c_i^2}} \tag{14}$$

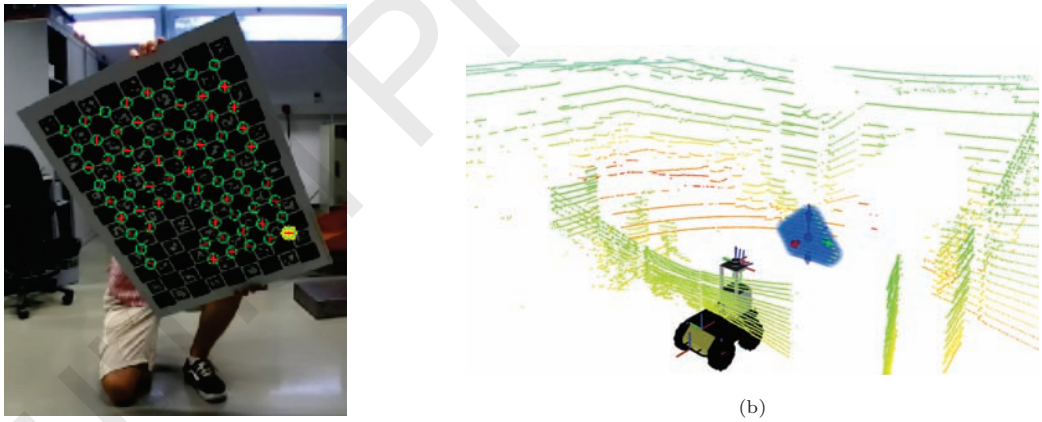
where $\boldsymbol{p}_j = [x_j, y_j, z_j]^T$ is the jth point on the cluster. With this, a point is considered as inlier if its distance to the plane D_{ij} is smaller than a given threshold $D_{threshold}$. The final set of inliers corresponds to the one found in the iteration i that gives the higher number of points belonging to the plane.

Figure 6 shows an example of the labeling procedure for cameras and 3D LiDARs proposed in ATOM.

It is worth noting that, our approach works with partial detections, as represented in the figure. This interactive data labeling procedure is showcased in `https://youtu.be/uNPIOCqxb5w`.

3.4. Collecting data

In most robotic systems, the data coming from the sensors is streamed at different frequencies. However, to compute the associations between the data of multiple sensors, temporal synchro-



(a) (b)

Figure 6: **(a)**: labeling image data using a charuco pattern; **(b)**: labeling 3D LiDAR data on pattern (solid blue points) using the semi-automatic proposed approach. Note that, our approach works with partial detections, i.e., collections of data where the pattern is not fully detected. This figure presents an example of an partial detection for each type of sensor. For the camera, only a portion of the corners of the pattern were detected. For the 3D LiDAR, the pattern is not fully observable due to the low vertical resolution. Even though, our approach is able to calibrate the system with these detections.

nization of the sensor data is required. Of course, this only becomes an issue when calibrating multi-sensor robotic systems. For now, the synchronization problem is solved trivially by collecting data (and the corresponding labels) at user defined moments in which the scene has remained static for a certain period of time. In static scenes, the problem of data de-synchronization is not observable, which warrants the assumption that for each captured collection the sensor data is "adequately" synchronized. This can be done using, for example, a tripod to held the pattern before collecting each snapshot of data (Rehder et al., 2016; Furgale et al., 2013). In this work, the problem is approached in a simpler way, where the pattern is hold by the user, as shown in Fig. 2, remaining static by sufficient amount of time to ensure the synchronization between all the sensors.

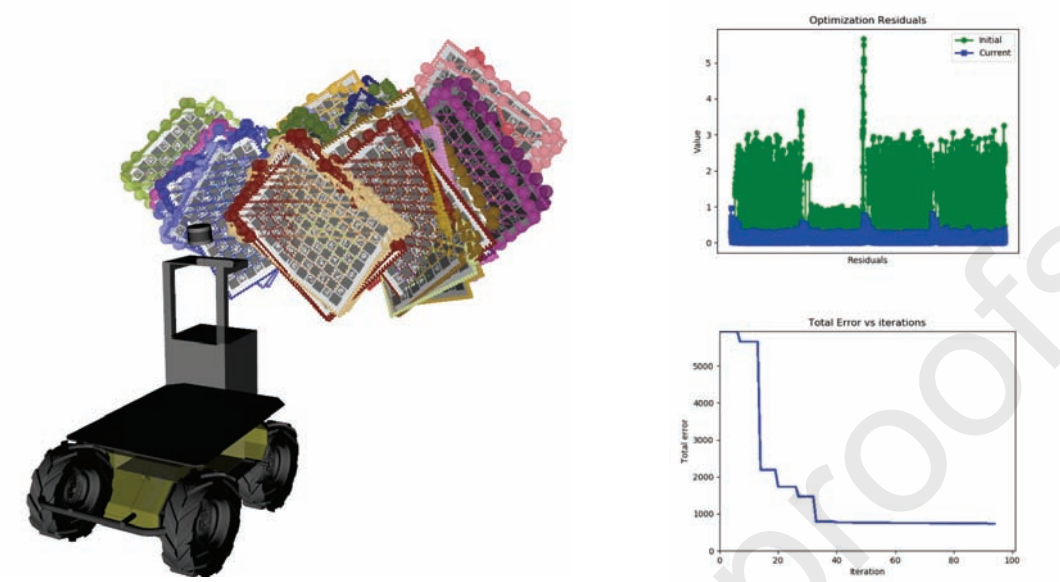
To save the scene data captured by all the sensors in the calibration system, the user can do it with just two mouse clicks on the interactive ROS-based tool (on RVIZ) developed. We refer to these recordings of data as *data collections*. Each one of them contains the values of all atomic transformations that exist in the system at a given timestamp, a copy of the robot configuration file, sensor data and labels, and high level information about each sensor, such as the topological transformation chain, extracted from the transformation tree. This information is stored in a *dataset file* that will be read by the optimization procedure afterwards. Also, a video showing the procedure for collecting data is provided for AgRob V16 calibration here `https://youtu.be/p7TuhSsRMcw`.

It should be pointed out that, the set of collections should contain as many different poses as possible. As such, collections should preferably have different distances and orientations w.r.t. the calibration pattern, so that the calibration returns more accurate results. This concern is common to the majority of calibration procedures.

3.5. Visualizing the optimization

The immediate visualization of the calibration is essential for several reasons: it provides the user the necessary data so that he can detect failures on the calibration, such as outlier data collections; it gives feedback about the cost function residuals minimization, which can serve to detect possible local minima, and to make sure that the optimization procedure is converging. Figure 7 shows the three main visualization features of ATOM.

Our calibration framework provides a simultaneous visualization of all the data collections, as well as immediate feedback of the alignment between ground-truth points and labeled points for optimization, images with the reprojection, robot meshes, the position of the reference frames,



(a) (b)

Figure 7: **(a)**: ROS calibration configuration - simultaneous visualization of all the collections of data and respective alignment between ground-truth points and labeled points; **(b)**: graphics representing the objective function minimization - individual residuals value, and the total error vs iterations.

etc. Also, optimization graphics are provided with residuals values and the total error for each iteration. This configuration is similar to the standard one which is used during the initial parameter estimation, the data labeling and collection, but contains a couple of key distinctions. As mentioned above, the calibration procedure uses a dataset file which contains information about each of the stored collections. These collections contain data gathered in a a set of sequential instants in time. The ROS calibration configuration publishes data from all collections simultaneously, as if those time instants were packed together and processed as if they had occurred all at the same time. Collisions in topic names and reference frames are avoided by adding a collection related prefix to each designation. Also, the original transformation tree is replicated for each collection. A video with an example of a calibration execution for AgRob V16 is provided here `https://youtu.be/HtTyTsWuMIQ`.

4. Results

To test and validate the performance of the proposed approach, an extensive evaluation procedure was developed. Our calibration framework, ATOM, was used to calibrate three configurations of the AgRob V16 sensing system. Two of them were pairwise calibrations between two cameras of a stereo system, and a single camera and a 3D LiDAR, which we denote as *ATOM pairwise*. The third was a calibration between all three sensors (two cameras and 3D LiDAR), which we call *ATOM full*, where results for particular pairs of sensors are obtained using a full calibration. In this procedure, three datasets were used, as represented in Tab. 1.

Two of them (*train-1*, *train-2*) were used for training, i.e., to perform the calibrations, and the third one (*test-3*) was used to test the calibration with specific metrics that will be detailed later on. The datasets contain incomplete and partial collections, i.e., collections where the pattern is not detected for all the sensors, and collections where the pattern is only partially visible, respectively. This section is divided in two parts: the evaluation of ATOM's performance against state-of-the-art approaches, such as OpenCV stereo calibration (Bradski, 2000), and ICP point cloud alignment (Besl & McKay, 1992); the characterization of ATOM w.r.t. several characteristics of the datasets, such as the number of incomplete/partial collections, and the accuracy of the initial guess. Table 2 makes a summary of the calibration experiments that were carried out.

OpenCV and the stereo camera factory calibration were used to get the camera-to-camera extrinsic calibration, and ICP was explored to get the camera-to-LiDAR calibration. This last calibration was obtained by the alignment of the 3D LiDAR point cloud, and the 3D point cloud

Table 1: Description of the datasets used for train and evaluation. Two datasets were used for training (calibration) and one for testing (evaluation). The datasets contain incomplete collections, i.e., collections were the pattern is not detected by at least one sensor, and partial collections, i.e., collections were the pattern is partially detected by at least one sensor.

Dataset	**Nr. Collections**			**Observations**
	Total	Incomplete	Partial	
train-1	42	5	38	Dataset contains incomplete collections.
train-2	56	0	24	The dataset does not contain the point cloud generated by the ZED camera.
test-3	15	0	8	Dataset with low number of collections, only used for test.

Table 2: Summary of the methods used and evaluated in the experiments.

Method	Calibration	Properties
OpenCV (Bradski, 2000)	pairwise	reprojection error, intrinsics calibration
Stereo camera factory calibration	pairwise	reprojection error, intrinsics calibrations
ICP average (Besl & McKay, 1992)	pairwise	reprojection error, average of result of all collections
ICP best (Besl & McKay, 1992)	pairwise	reprojection error, best result of all collections
ATOM pairwise [this paper]	pairwise	reprojection error, angle-axis, intrinsics calibration
ATOM full [this paper]	full calibration	reprojection error, angle-axis, intrinsics calibration

provided by the stereo camera software development kit. Two versions of ICP were used as camera-to-LiDAR extrinsic calibration: the one corresponding to the collection where the fitting of point clouds was more accurate, and the average of the transformations obtained in all collections. Finally, as referenced before, ATOM was calibrated both in pairwise and full modes, and both approaches are evaluated.

4.1. Methodology

One of the key characteristics of our evaluation procedure is the use of separate datasets to perform the calibration and generate the results. As discussed in Sec. 3, ATOM provides a data collection procedure, where datasets are generated and visualized on RVIZ . Datasets are composed of several collections, each one containing data of all the sensors, initial atomic transformations, pattern labelled points, and other information. To evaluate our calibration framework, two datasets where initially collected - *train-1* and *train-2*. Then, three calibrations were executed over each one of the datasets, two pairwise and one using all the sensors to be calibrated in AgRob V16 system. These calibrations generate a `json` file similar to the one generated at the end of the data collection procedure, but with the calibrated atomic transformations. After obtaining all these calibration configurations, a third dataset was recorded - *test-3*. It was used to evaluate the accuracy of the calibrations obtained. Using the metrics that will be described later on, the chain of transformations of each calibration was loaded and used to compute errors using the labelled data of the test dataset. In this way, possible influence of using the same data to calibrate and test is eliminated, and a more rigorous evaluation is achieved.

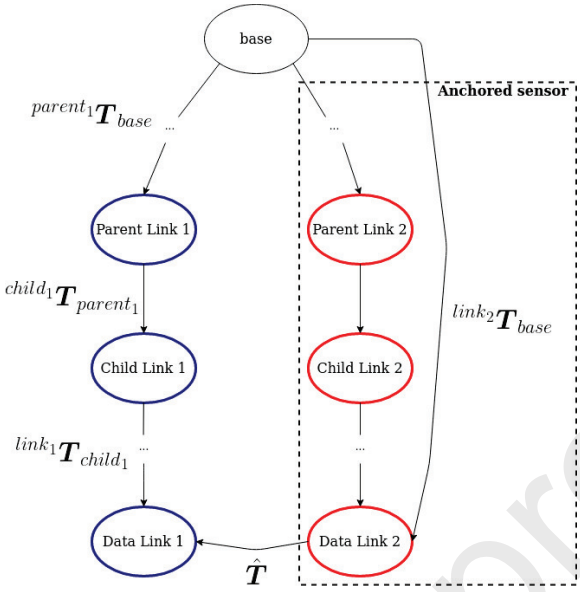


Figure 8: Chain of transformations representing the generic configuration to extract an atomic transformation from a sensor to sensor calibration. Preserving the chain of transformation of the anchored sensor, and taking into account the sensor to sensor calibration $\hat{\boldsymbol{T}}$, we recover the atomic transformation $^{child_1}\boldsymbol{T}_{parent_1}$.

Unlike ATOM, both OpenCV and ICP perform sensor-to-sensor calibration, instead of calibrating atomic transformations without changing the topology of the chain of transformations. Thus, to evaluate these methods in the same way ATOM is evaluated, the atomic transformations set to be calibrated had to be recovered from the sensor-to-sensor calibrations. This problem is formulated in Fig. 8.

Let $^{link_2}\boldsymbol{T}_{base}$ be the entire chain of transformations from the base link to the data link of the anchored sensor, $\hat{\boldsymbol{T}}$ be the sensor to sensor calibration obtained, $^{parent_1}\boldsymbol{T}_{base}$ be the chain of transformations from the base link to the parent link of the atomic transformation to be calibrated $^{child_1}\boldsymbol{T}_{parent_1}$, and $^{link_1}\boldsymbol{T}_{child_1}$ the chain of transformation from the atomic transformation child link, to the non-anchored sensor data link. The entire chain of transformations relationships can be formulated as follows:

$$\hat{\boldsymbol{T}} \cdot {}^{link_2}\boldsymbol{T}_{base} = {}^{link_1}\boldsymbol{T}_{child_1} \cdot {}^{child_1}\boldsymbol{T}_{parent_1} \cdot {}^{parent_1}\boldsymbol{T}_{base}. \tag{15}$$

So, from (15), we can extract the atomic transformation of the non-anchored sensor as follows:

$$^{child_1}\boldsymbol{T}_{parent_1} = (^{link_1}\boldsymbol{T}_{child_1})^{-1} \cdot \hat{\boldsymbol{T}} \cdot {}^{link_2}\boldsymbol{T}_{base} \cdot (^{parent_1}\boldsymbol{T}_{base})^{-1}. \tag{16}$$

The advantage of this approach is that it is generic. For example, from OpenCV, a camera to camera metric is obtained. Thus, the procedure consists in anchoring one of the cameras, and use the obtained transformation to recover the atomic transformation marked for calibration in the original ATOM configuration. In the same way, ICP provides a camera to LiDAR calibration. Once again, we anchor one of these sensors, and apply the exact same routine to extract the non-anchored sensor atomic transformation to be calibrated. In this way, we are able to obtain a calibrated system without changing the initial chain topology, which allows the direct comparison of these state-of-the-art approaches with ATOM, using exactly the same metrics. These metrics are described in the next section.

4.2. Metrics

To evaluate the camera to camera calibration performance, the methodology used is based on three different metrics: the mean rotation error (rad), the mean translation error (m), and the reprojection error (px). To compute the reprojection error, the idea is to use the calibration result to project the detected pattern corners of one camera image into the image of the second calibrated camera image and compare the projected pixel coordinates with the ground truth pattern corners coordinates in the image of the anchored camera. To transform pixels from one camera into another, we start from projecting the 3D world coordinates of the pattern corners into the image of a camera, using (6)-(7). Since the 3D pattern corners are defined in the local pattern reference frame, they all lie in the plane $z = 0$. Thus, (6)-(7) can be simplified to the following:

$$\boldsymbol{p}^{cam} = \boldsymbol{K} \cdot {}^{cam}\boldsymbol{T}'_p \cdot \boldsymbol{p}^{p'}, \tag{17}$$

where $^{cam}\boldsymbol{T}'_p$ is a portion of the matrix $^{cam}\boldsymbol{T}_w \cdot {}^{w}\boldsymbol{T}_p$, without the component z of the rotation, as follows:

$$^{cam}\boldsymbol{T}'_p = \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix}, \tag{18}$$

and $\boldsymbol{p}^{p'}$ is the pattern corner, represented as a vector in its homogeneous form, without the z component, i.e., $\boldsymbol{p}^{p'} = \begin{bmatrix} x & y & 1 \end{bmatrix}^T$. Using the fact that the 3D coordinates of the pattern's corners are the same for both cameras, (17) can be applied to the two of them, so that we can find a relation between both expressions. This resulted in the following formulation:

$$\boldsymbol{p}^{cam2} = \boldsymbol{K}^{cam2} \cdot {}^{cam2}\boldsymbol{T}'_p \cdot \left({}^{cam1}\boldsymbol{T}'_p\right)^{-1} \cdot \left(\boldsymbol{K}^{cam1}\right)^{-1} \cdot \boldsymbol{p}^{cam1}, \tag{19}$$

where $cam1$ and $cam2$ refer to the cameras that were calibrated. This formulation provides the relationship between pixel coordinates of the pattern corners in both camera images. However, (19) requires the camera to pattern transformation matrix for both cameras. This can be a problem since, some approaches, unlike ATOM, do not estimate the camera to pattern transformation while performing the camera to camera calibration. In addition to this, ATOM estimates these transformations for a training dataset. So, the usage of a test dataset to evaluate all the frameworks, denies the use of the estimated pattern pose from ATOM. To overcome this, the pattern pose w.r.t. one of the cameras ${}^{cam1}T_p$ is computed using the PnP algorithm. Then, using the output json file from each calibration, we recover the camera to camera transformation ${}^{cam1}\boldsymbol{T}_{cam2}$, through the chain of transformations. In this manner, it is possible to determine the transformation of the other camera to the pattern, as follows:

$${}^{cam2}\boldsymbol{T}_p = \left({}^{cam1}\boldsymbol{T}_{cam2}\right)^{-1} \cdot {}^{cam1}\boldsymbol{T}_p. \tag{20}$$

From this expression, we can derive ${}^{cam2}\boldsymbol{T}'_p$ and ${}^{cam1}\boldsymbol{T}'_p$, and successfully project pixels from one image into the other. With this information, the reprojection error is computed as follows:

$$\boldsymbol{e}_{xy} = \boldsymbol{p}^{projected} - \boldsymbol{p}^{expected}. \tag{21}$$

From (21), the error can be decomposed in its x and y components. Also, considering the reprojection error for all the N collections, the root mean square error is calculted as follows:

$$e_{rms} = \sqrt{\frac{1}{N}\sum \boldsymbol{e}_{xy}^2}. \tag{22}$$

Figure 9 illustrates a resulting corner reprojection from one camera into the other using the ATOM full calibration.

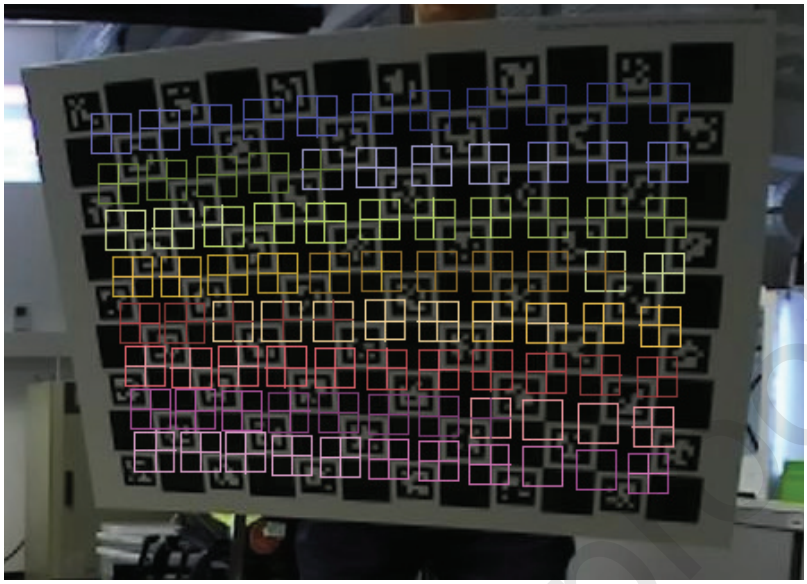


Figure 9: Camera reprojection error from one camera image into the other. Squares represent the expected corner pixel coordinates, and crosses the projected result.

For the calculation of the mean rotation and translation errors to evaluate the camera to camera calibration, we consider the following observation: the chain of transformations from the base link to the pattern reference frame that passes from each one of the calibrated cameras should be equal. This happens since the calibration pattern pose in reference with the base link is fixed, and any chain of transformations that link these two referentials should represent the same spatial relationship. Thus, the difference in rotation and translation can be quantified, assessing the inequality between the two chains of transformation. Once again, this formulation requires the pattern pose w.r.t. each one of the cameras, that is again extracted solving the PnP problem. With this information, we can state that

$$\begin{bmatrix} {}^{base}\boldsymbol{R}_{cam1} & {}^{base}\boldsymbol{t}_{cam1} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{cam1}\boldsymbol{R}_{p} & {}^{cam1}\boldsymbol{t}_{p} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^{base}\boldsymbol{R}_{cam2} & {}^{base}\boldsymbol{t}_{cam2} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{cam2}\boldsymbol{R}_{p} & {}^{cam2}\boldsymbol{t}_{p} \\ 0 & 1 \end{bmatrix}. \tag{23}$$

Now, we can define the rotation and translation difference as

$$\Delta\boldsymbol{R} = \left({}^{base}\boldsymbol{R}_{cam1} \cdot {}^{cam1}\boldsymbol{R}_{p}\right)^{-1} \cdot {}^{base}\boldsymbol{R}_{cam2} \cdot {}^{cam2}\boldsymbol{R}_{p} \tag{24}$$

$$\Delta \boldsymbol{t} = {}^{base}\boldsymbol{R}_{cam1} \cdot {}^{cam1}\boldsymbol{t}_p + {}^{base}\boldsymbol{t}_{cam1} - {}^{base}\boldsymbol{R}_{cam2} \cdot {}^{cam2}\boldsymbol{t}_p - {}^{base}\boldsymbol{t}_{cam2} \tag{25}$$

Finally, we can define the mean rotation error as

$$e_R = \frac{1}{N} \sum_i ||\texttt{angle}(\Delta \boldsymbol{R}_i)||, \tag{26}$$

where `angle` is the angle-axis representation of the rotation, and the mean translation error as

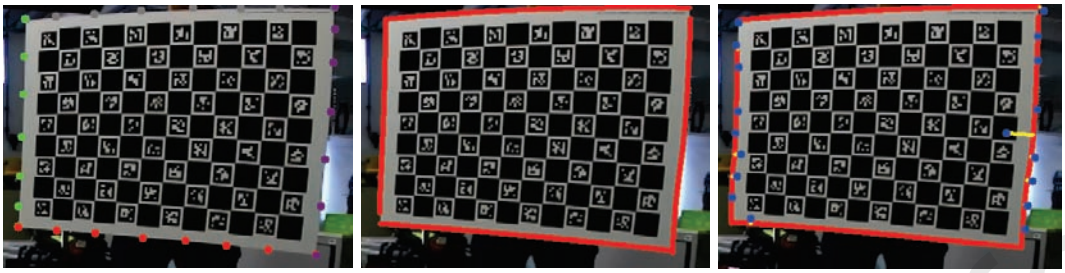
$$e_t = \frac{1}{N} \sum_i ||\Delta \boldsymbol{t}_i||. \tag{27}$$

To evaluate the 3D LiDAR to camera calibration, we used the reprojection error (px) and its corresponding root mean square error (px) considering all the test collections N. In this case, the mean rotation and translation errors were not used due to the difficulty of estimating the pattern pose w.r.t. the LiDAR sensor with precision. The process of calculating the reprojection error to evaluate the camera to LiDAR calibration consists in three main steps:

1. Label the pixels that belong to the boundaries of the pattern in the image.
2. Reproject the pattern boundary points in the LiDAR's referential frame (detailed in Sec. 2.1) $\boldsymbol{p}^{boardlimit}$ into the image.
3. Compute root mean square between labeled and projected points.

Figure 10 shows these steps.

An annotation tool was developed to perform the labelling. This tool allows the user to manually annotate individual points corresponding to four classes, each one representing one side of the pattern in each image. Then, in order to account for the image distortion, we approximate each one of the pattern sides by a polynomial, fitting the labelled points. In this step, a simple linear regression would not suffice because images have distortion which transforms straight ines into curves. So, a polynomial is more suitable for modeling this phenomenon. Figures 10a and 10b show these two steps. After having the annotations for all the images of the camera to be calibrated in the test dataset (test-3), the reprojection error is calculated. To do so, the 3D LiDAR labelled points that belong to the pattern boundaries are reprojected into the image using (6)-(9). So, for each collection, the error between each projected point and the closest ground truth point belonging to one of the polynomial curves is calculated as in (21). Figure 10c shows an example of the



(a) (b) (c)

Figure 10: 3D LiDAR to camera reprojection error metric calculation. **(a)** represents the annotation procedure, where four classes are annotated, each one representing a single side of the pattern; **(b)** red curves represent the approximation of each one of the classes by a polynomial function, in order to account for the distortion in the image; **(c)** reprojection error calculation - blue dots represent the 3D LiDAR pattern boundary reprojected points and yellow lines the difference of each one of the points with the labelled ground truth.

reprojection result and the corresponding error for each point. Considering the reprojection errors calculated for each one of the N collections, the root mean square error is also calculated using (22).

It should be emphasised that, all of these metrics are publicly available, and are integrated in the ATOM software framework. A specific package called *ATOM evaluation* was created and can be easily used by the user to evaluate the calibrations performed.

4.3. Evaluation

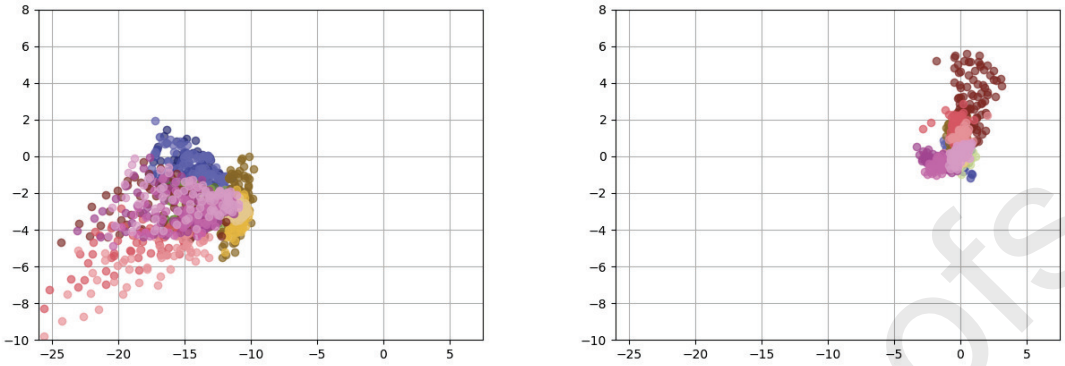
The evaluation procedure applies the previously described metrics to compare ATOM with state-of-the-art calibration methods. Note that these state-of-art methods are pairwise and as such not able to calibrate the entire system simultaneously. The comparison with ATOM, which is a general, full calibration approach, against specialized pairwise methods is not entirely fair. However, since the nature of all the metrics used in the evaluation is also pairwise, it is ATOM that is at a disadvantage in comparison with the other methods. For the camera-to-camera calibration scenario, two versions of ATOM were calibrated in two diffeernt training datasets, and evaluated in the same test dataset. The first is a pairwise calibration between both cameras, and the second a full calibration of the entire AgRob V16 system, with the same two cameras and a 3D LiDAR. It is worth noting that, the calibrations performed consider each camera intrinsic parameters, as well as the pairwise extrinsic calibration between them. To compare ATOM with the state of the

Table 3: Performance comparison of methods for camera to camera calibration.

Method	**Train dataset**	$e_{\boldsymbol{R}}$ (rad)	$e_{\boldsymbol{t}}$ (m)	e_x (px)	e_y (px)	e_{rms} (px)
OpenCV (Bradski, 2000)	train-1	Not able to calibrate due to partial pattern detections.				
ATOM pairwise		0.009	0.003	0.551 ± 0.800	0.780 ± 1.090	1.049
ATOM full		0.008	0.003	0.547 ± 0.759	0.638 ± 1.034	**0.974**
OpenCV (Bradski, 2000)	train-2	0.006	0.003	0.582 ± 0.648	0.622 ± 0.966	**0.863**
ATOM pairwise		0.010	0.006	0.655 ± 1.055	0.677 ± 0.982	1.020
ATOM full		0.008	0.005	0.594 ± 0.912	0.696 ± 1.033	0.974
ZED's factory calibration	-	0.007	0.006	2.220 ± 0.765	0.486 ± 0.823	1.757

art, the OpenCV stereo calibration toolbox (Bradski, 2000) was used to calibrate exactly the same configuration. Additionally, the factory's intrinsic and extrinsic calibrations were evaluated. Table 3 summarizes all these experiments.

Starting by the analysis of ATOM pairwise and ATOM full, we can see that both versions present similar performances, with marginal differences with respect to all the metrics calculated. For example, for the train-1 dataset, we can see a root mean square error difference of 0.075 px and for train-2 0.046 px. Thus, we can conclude that, ATOM allows to optimize an entire robotic system without a significant loss of performance, when comparing with a specific pairwise calibration between the two sensors of interest. In what concerns OpenCV, Tab. 3 shows that, for the train-1 dataset, it is not able to calibrate. This happens since this framework requires collections where all the pattern corners are detected, i.e., non partial detections. So, since the majority of the collections present in this dataset are partial, OpenCV is not able to calibrate. This is a limitation since, to accomplish a dataset without partial collections, its variety can be limited due to the impossibility of collecting, e.g., collections with the pattern far away from the cameras. On the other hand, for the train-2 dataset, OpenCV achieves the smaller reprojection root mean square error. In this, the number of partial collections is low, and OpenCV, a specialized pairwise calibrator for cameras, performs an accurate intrinsic and extrinsic calibration. Even so, ATOM full shows errors only slightly higher than OpenCV for this dataset, showing that it is capable of achieving a state-of-the-art performance, even considering a non pairwise approach. Concerning the camera factory calibration, it is clear that it presents the less accurate calibration. This can be explained since the calibration the same for all the equipments. Finally, to analyse the impact



(a) (b)

Figure 11: Reprojection error dispersion for the camera to camera calibration using ATOM full configuration. Each color represents the error associated with one individual collection. **(a)** is the representation of this error before calibrating (using the initial guess), and **(b)** after the calibration.

of the ATOM's calibration, Fig. 11 shows the dispersion of the reprejection error per collection, before and after calibrating with ATOM full.

As expected, the dispersion of the error before calibrating is higher in almost all collections. On the contrary, after calibrating the cameras with ATOM full, the dispersion drastically reduces, with the exception of one collection (represented in brown). This collection can represent a degenerate of data collection, due to, for example, de-synchronization of the data from from the sensors.

In order to evaluate the camera to LiDAR calibration, ATOM was used to calibrate both modalities in three different manners: two ATOM pairwise versions, one between the LiDAR and each camera, and the ATOM full version that comprises all three sensors. To compare our approach with the state-of-the-art, ICP (Besl & McKay, 1992) was used to calibrate the left camera and the LiDAR in two different ways: one considering the average of the calibration obtained in all the collections, and other considering only the collection that presents the best alignment between the two point clouds. Note that, ICP only calibrates the left camera w.r.t. the LiDAR since the stereo camera point cloud extracted directly using the manufacture's API is defined in this camera referential. Table 4 summarizes all this information.

Similarly to the camera-to-camera case, here we can verify that ATOM pairwise and ATOM full result in a similar calibration performance, with marginal reprojection error differences. So, once

Table 4: Performance comparison of methods for camera to 3D LiDAR calibration.

Method	**Type**	**Train dataset**	e_x (px)	e_y (px)	e_{rms} (px)
ICP average (Besl & McKay, 1992)	left camera - 3D LiDAR	train-1	47.210 ± 31.374	19.058 ± 28.233	44.307
ICP best (Besl & McKay, 1992)	left camera - 3D LiDAR		9.111 ± 11.950	2.625 ± 7.967	10.492
ATOM pairwise	right camera - 3D LiDAR		3.054 ± 4.727	1.031 ± 2.689	3.869
ATOM pairwise	left camera - 3D LiDAR		3.648 ± 4.846	1.260 ± 2.869	4.101
ATOM full	right camera - 3D LiDAR		3.351 ± 4.874	0.950 ± 2.279	**3.811**
ATOM full	left camera - 3D LiDAR		3.398 ± 4.923	1.100 ± 2.602	3.942
ICP average (Besl & McKay, 1992)	left camera - 3D LiDAR	train-2	-	-	-
ICP best (Besl & McKay, 1992)	left camera - 3D LiDAR		-	-	-
ATOM pairwise	right camera - 3D LiDAR		7.574 ± 6.393	1.776 ± 3.181	6.715
ATOM pairwise	left camera - 3D LiDAR		7.560 ± 5.535	1.619 ± 2.795	**6.432**
ATOM full	right camera - 3D LiDAR		7.702 ± 5.441	1.648 ± 2.781	6.537
ATOM full	left camera - 3D LiDAR		8.117 ± 5.692	1.687 ± 2.838	6.765

again, this leads to the conclusion that ATOM full can be used to calibrate all the robotic system without any significant loss of performance while evaluating calibrations between pairs of sensors. In this set of tests, a consistent decrease of performance of all the calibration configurations from train-1 to train-2 dataset is observed. This consistency can be caused by synchronization errors between sensors while collecting the calibration data. Looking for the ICP performance on train-1 dataset, firstly, we can conclude that the ICP average is highly affected by outliers, i.e., collections where the calibration fails. This can be inferred by the high standard deviations present in the x and y reprojection error compoenents. ICP best, despite being significantly less accurate than ATOM, presents a better performance than ICP average. The overall bad performance of ICP can be explained by the difficulty of aligning a dense point cloud (provided by the stereo camera), and a sparse one (provided by the laser). It is worth noting that, the train-2 dataset does not contain the stereo camera point cloud, so here ICP can not be used for calibration. To have a visual perception of the ATOM full performance on the calibration of the left camera and the LiDAR, Fig. 12 shows the reprojection of the LiDAR 3D points in the left camera image.

In this Fig., color represents the points depth. Here, the transitions of the objects can be sharply observed, which is a good indicator for the calibration performance.

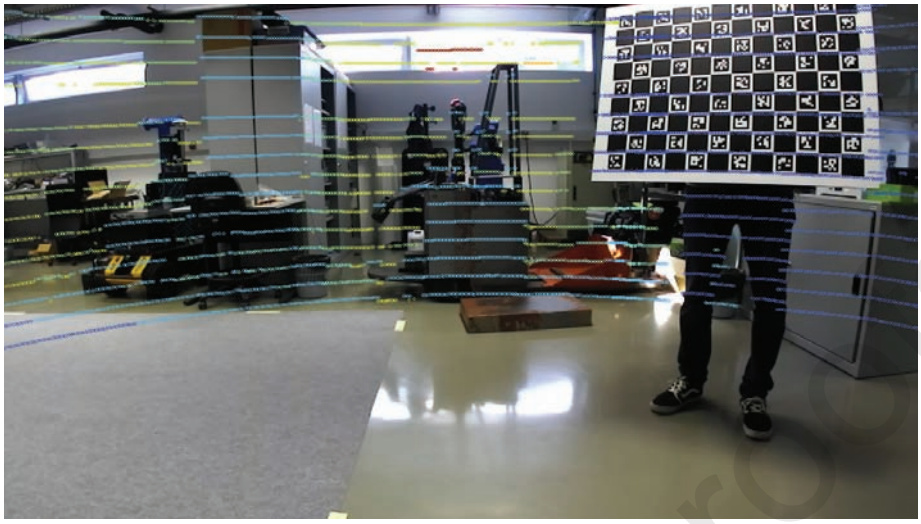


Figure 12: Point cloud projection into the left camera image using the 3D LiDAR to camera calibration. The color represents the points depth. It is worth noting that, this procedure was done using the ATOM full calibration result, with a collection from the test dataset, just like in all the evaluation pipeline. If the system is accuratly calibrated, changes in point's color, which denotes a variation of the measured range, should coincide with transitions between far and near objects in the image.

4.4. Impact of the number of collections used for training

One of the major questions in general for calibration procedures is the minimum amount of data required to calibrate sensors with precision. In this section we propose an evaluation of ATOM full calibration using different numbers of training collections. The calibration of the three combinations of sensors is evaluated for five different levels of collections used. Table 5 presents the results obtained for each configuration.

Starting by the analysis of the camera-to-camera calibration, here we can see that the increase of the number of collections leads to a increase in performance. Using a single collection, as expected, results in higher reprojection, rotation and translation errors. While increasing the number of training collections, the performance increases, with the best performance being observed with the maximum number of collections. Looking at the performance of the calibration of the LiDAR with both cameras, we can see that, as expected using a single collection also results in a higher reprojection error. However, in this case, this difference is not significant, and while increasing the number of collections, the performance saturates. Thus, we can conclude that, the increase

Table 5: Impact of the number of collections in ATOM's full calibration performance. The dashed entries correspond to results not generated since, for the camera-to-LiDAR case, the mean rotation and translation errors are not available.

Type	Nr. Collections	$\mathbf{e}_{\boldsymbol{R}}$ (rad)	$\mathbf{e}_{\boldsymbol{t}}$ (rad)	$\mathbf{e}_x$ (px)	$\mathbf{e}_y$ (px)	$\mathbf{e}_{rms}$ (px)
camera - camera	1	0.051	0.053	8.205 ± 7.201	16.089 ± 3.346	13.534
	5	0.017	0.016	3.540 ± 4.575	1.108 ± 1.449	4.233
	10	0.009	0.004	1.377 ± 1.318	0.932 ± 0.991	1.645
	20	0.008	0.003	0.515 ± 0.719	0.658 ± 1.056	0.976
	30	0.008	0.003	0.547 ± 0.759	0.638 ± 1.034	**0.974**
right camera - 3D LiDAR	1	-	-	4.348 ± 6.314	1.936 ± 4.431	5.487
	5	-	-	4.202 ± 5.179	1.144 ± 2.342	4.466
	10	-	-	3.305 ± 4.742	0.984 ± 2.387	3.820
	20	-	-	3.355 ± 4.744	1.005 ± 2.412	**3.774**
	30	-	-	3.352 ± 4.874	0.950 ± 2.279	3.811
left camera - 3D LiDAR	1	-	-	5.126 ± 6.686	1.527 ± 3.435	5.566
	5	-	-	3.381 ± 4.447	1.058 ± 2.503	3.730
	10	-	-	2.937 ± 4.430	1.115 ± 2.791	**3.712**
	20	-	-	3.411 ± 4.887	1.258 ± 2.959	4.046
	30	-	-	3.398 ± 4.924	1.100 ± 2.602	3.942

of the number of collections has a positive impact in the final performance of all the calibration configurations. However, the camera-to-camera calibration is more sensible to the lower number of collections than the camera-to-LiDAR calibration.

4.5. Impact of the number of incomplete collections used for training

The proposed calibration framework, ATOM, supports collections where the pattern is not detected by all the sensors in the calibration system. For example, suppose that the pattern is, for a specific collection, is viewed by the right camera and the LiDAR but not by the left camera. The current section intends to evaluate the impact of this type of collections, and conclude if the the presence of incomplete collections has, or not, correlation with changes on ATOM's performance. Table 6 summarizes the results obtained for three sensor configurations with four different values of incomplete collections, maintaining the same number of training collections.

ATOM can deal with incomplete collections, as long as the pattern is detected by at least one sensor. If not, the calibration system can not compute any residual, and that collection must be

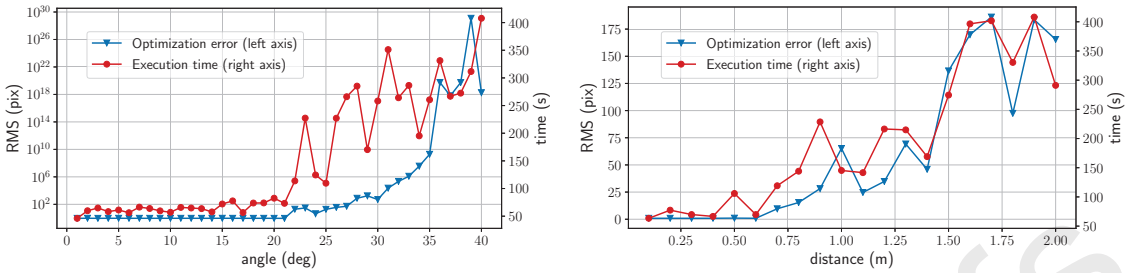
Table 6: Impact of the number of incomplete collections in ATOM full calibration performance. The dashed entries correspond to results not generated since, for the camera-to-LiDAR case, the mean rotation and translation errors are not available.

Type	**Nr. Collections**		$\mathbf{e_R}$ **(rad)**	$\mathbf{e_R}$ **(rad)**	$\mathbf{e}_x$ **(px)**	$\mathbf{e}_x$ **(px)**	$\mathbf{e}_{rms}$ **(px)**
	Complete	Incomplete					
camera - camera	10	0	0.011	0.011	1.159 ± 1.479	0.848 ± 1.182	1.457
	10	2	0.006	0.003	0.821 ± 0.711	0.606 ± 0.900	**1.031**
	10	4	0.009	0.006	1.680 ± 2.001	0.784 ± 0.945	2.017
	10	5	0.010	0.005	0.778 ± 0.990	0.641 ± 0.908	1.076
right camera - 3D LiDAR	10	0	-	-	5.104 ± 6.517	1.370 ± 3.085	5.496
	10	2	-	-	3.272 ± 4.800	1.068 ± 2.634	**3.920**
	10	4	-	-	3.414 ± 4.934	1.040 ± 2.515	3.964
	10	5	-	-	3.734 ± 4.956	1.111 ± 2.540	4.069
left camera - 3D LiDAR	10	0	-	-	5.113 ± 6.479	1.535 ± 3.430	5.576
	10	2	-	-	3.112 ± 4.665	1.192 ± 2.970	3.930
	10	4	-	-	2.995 ± 4.541	1.137 ± 2.886	**3.849**
	10	5	-	-	3.720 ± 5.012	1.274 ± 2.907	4.169

discarded. On the other hand, if, a single sensor does not detect the pattern for a specific collection, this leads to a reduction of the number of residuals used on the optimization procedure. This being said, results do not show any correlation between the increase of the number of incomplete collections and the performance of ATOM. So, this leads to the conclusion that ATOM can deal with the decrease of the number of residuals (as long as sufficient number of collections are provided). This conclusion consistent with the one taken from Tab. 5, where it was shown that, ATOM can calibrate accurately for a reasonable low number of collections.

4.6. Impact of the accuracy of the initial estimate

The initial estimate (or initial guess) of the calibration parameters has an impact in the outcome of the optimization. A good initial estimate provides a sufficient approximation that allows the optimization process to find the optimal solution that best represents the real calibration of the system. In turn, a inaccurate estimate may lead the optimization process to an unrecoverable state where it is not possible to achieve the optimal solution. This is known as the problem of local minima.



(a) (b)

Figure 13: Impact of the initial angle (a) and distance (b) estimation error on the optimization error and execution time. The considered optimization error is the *root means squared* (RMS) error.

In this section, we are interested in assessing the robustness of ATOM to the accuracy of the initial estimate. More specifically, the angle and distance error to the optimal state that our method can handle and the additional execution times that result from said errors. To characterize the robustness to the angle error, we start with the optimal angle and then add the angle error to the the Euler components of the rotation. The sign of the error is provided by a fair binomial sampling (Girshick et al., 2006). The tolerance to the distance error is found by sampling an uniform offset from the optimal positions that sits on a sphere with a radius equal to the distance error. Because random sampling is used, we run the experiment for each error 10 times and the reported values are the mean of the runs. By increasing the error in several steps, we can pinpoint the error at which our optimization process will fail. Note that the error is applied to all pose parameters that are being estimated. The results are presented in Figure 13.

The obtained results show that our optimization process can handle an angle error of approximate 20 degrees and a distance error of 0.5 meters, for each sensor. In our opinion, these errors provide a sufficient margin of tolerance for a practical usage of our manual procedure for initial parameter estimation, described in section 3.2. The execution times are strictly related to the convergence of the optimization. Inside the margin of tolerance, the execution times are mostly constant (with some fluctuations). This means the optimization process adequately handles the imposed error with a proper convergence.

5. Conclusions

This paper solves the problem of camera-to-LiDAR calibration using the optimization of atomic transformations. To do so, this work formulates the calibration as a Bundle Adjustment problem, minimizing the reprojection error of sensors that can have different modalities. Our approach, ATOM, provides several advantages when compared with the current state-of-the-art: (i) it offers a framework to simultaneously calibrate any number of sensors; (ii) it improves the optimization of different sensor modalities by introducing the multi-modal normalization. (iii) it maintains the topology of the input transformation tree; (iv) it supports incomplete and partial collections of data, which makes the detection procedure more flexible and robust; (v) it uses a common calibration pattern, which generalizes the approach; (vi) it has seamless integration with ROS, setting a complete framework for camera to LIDAR calibration.

Results show that the proposed approach presents similar performance in comparison with the state-of-the-art, even calibrating the entire robotic system simultaneously. These results demonstrate that ATOM can achieve the same performance of specialized methods in pairwise calibration between specific sensors, while running a complete calibration with multiple sensors of different modalities. Furthermore, our framework proved to be robust to inaccurate initial guesses and small number of collections. Finally, the use of a generic calibration pattern constitutes a major advance since, in the current state-of-the-art, many approaches use built in-house patterns.

Future work aims to test ATOM in more advanced robotic systems, with multiple 3D LiDARs. Additionally, the problem of data synchronization while collecting data for calibration will be addressed thought the use of simple concepts such as data interpolation, and more advanced ones, such as generative adversarial networks to generate synchronized sensor data.

Acknowledgements

This Research was funded by National Funds through the FCT—Foundation for Science and Technology, in the context of the project UIDB/00127/2020.

References

Agarwal, S., Snavely, N., Seitz, S. M., & Szeliski, R. (2010). Bundle adjustment in the large. In K. Daniilidis, P. Maragos, & N. Paragios (Eds.), *Computer Vision – ECCV 2010* (pp. 29–42). Berlin, Heidelberg: Springer Berlin Heidelberg.

de Aguiar, A. S. P., dos Santos, F. B. N., dos Santos, L. C. F., de Jesus Filipe, V. M., & de Sousa, A. J. M. (2020). Vineyard trunk detection using deep learning – an experimental device benchmark. *Computers and Electronics in Agriculture*, *175*, 105535. URL: https://doi.org/10.1016/j.compag.2020.105535. doi:10.1016/j.compag.2020.105535.

Álvarez, S., Llorca, D., & Sotelo, M. (2014). Hierarchical camera auto-calibration for traffic surveillance systems. *Expert Systems with Applications*, *41*, 1532–1542. URL: https://doi.org/10.1016/j.eswa.2013.08.050. doi:10.1016/j.eswa.2013.08.050.

Badue, C., Guidolini, R., Carneiro, R. V., Azevedo, P., Cardoso, V. B., Forechi, A., Jesus, L., Berriel, R., Paixão, T. M., Mutz, F., de Paula Veronese, L., Oliveira-Santos, T., & De Souza, A. F. (2021). Self-driving cars: A survey. *Expert Systems with Applications*, *165*, 113816. URL: http://www.sciencedirect.com/science/article/pii/S095741742030628X. doi:https://doi.org/10.1016/j.eswa.2020.113816.

Besl, P., & McKay, N. D. (1992). A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *14*, 239–256. URL: https://doi.org/10.1109/34.121791. doi:10.1109/34.121791.

Bradski, G. (2000). The opencv library. *Dr Dobb's J. Software Tools*, *25*, 120–125. URL: https://ci.nii.ac.jp/naid/10028167478/en/.

Czyzewski, M. A. (2017). An extremely efficient chess-board detection for non-trivial photos. *ArXiv*, *abs/1708.03898*.

Dhall, A., Chelani, K., Radhakrishnan, V., & Krishna, K. M. (2017). Lidar-camera calibration using 3d-3d point correspondences. arXiv:arXiv:1705.09785.

Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, *13*, 99–110. URL: https://doi.org/10.1109/mra.2006.1638022. doi:10.1109/mra.2006.1638022.

Fabbri, R., Giblin, P., & Kimia, B. (2020). Camera pose estimation using first-order curve differential geometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (pp. 1–1). URL: https://doi.org/10.1109/tpami.2020.2985310. doi:10.1109/tpami.2020.2985310.

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*, 381–395. URL: https://doi.org/10.1145/358669.358692. doi:10.1145/358669.358692.

Foote, T. (2013). tf: The transform library. In *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*. IEEE. URL: https://doi.org/10.1109/tepra.2013.6556373. doi:10.1109/tepra.2013.6556373.

Fremont, V., & Bonnifait, P. (2008). Extrinsic calibration between a multi-layer lidar and a camera. In *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. IEEE. URL: https://doi.org/10.1109/mfi.2008.4648067. doi:10.1109/mfi.2008.4648067.

Furgale, P., Rehder, J., & Siegwart, R. (2013). Unified temporal and spatial calibration for multi-sensor systems. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. URL: https://doi.org/10.1109/iros.2013.6696514. doi:10.1109/iros.2013.6696514.

Gao, X.-S., Hou, X.-R., Tang, J., & Cheng, H.-F. (2003). Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*, 930–943. URL: https://doi.org/10.1109/tpami.2003.1217599. doi:10.1109/tpami.2003.1217599.

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F., & Medina-Carnicer, R. (2016). Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognition*, *51*, 481–491. URL: https://doi.org/10.1016/j.patcog.2015.09.023. doi:10.1016/j.patcog.2015.09.023.

Girshick, M. A., Mosteller, F., & Savage, L. J. (2006). Unbiased estimates for certain binomial sampling problems with applications. In S. E. Fienberg, & D. C. Hoaglin (Eds.), *Selected Papers of Frederick Mosteller* (pp. 57–68). New York, NY: Springer New York. URL: https://doi.org/10.1007/978-0-387-44956-2_3. doi:10.1007/978-0-387-44956-2_3.

Guindel, C., Beltran, J., Martin, D., & Garcia, F. (2017a). Automatic extrinsic calibration for lidar-stereo vehicle sensor setups. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. URL: https://doi.org/10.1109/itsc.2017.8317829. doi:10.1109/itsc.2017.8317829.

Guindel, C., Beltran, J., Martin, D., & Garcia, F. (2017b). Automatic extrinsic calibration for lidar-stereo vehicle sensor setups. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. URL: https://doi.org/10.1109/itsc.2017.8317829. doi:10.1109/itsc.2017.8317829.

Hornegger, J., & Tomasi, C. (1999). Representation issues in the ML estimation of camera motion. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*. IEEE. URL: https://doi.org/10.1109/iccv.1999.791285. doi:10.1109/iccv.1999.791285.

Hu, D., DeTone, D., & Malisiewicz, T. (2019). Deep ChArUco: Dark ChArUco marker pose estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. URL: https://doi.org/10.1109/cvpr.2019.00863. doi:10.1109/cvpr.2019.00863.

Huang, J.-K., & Grizzle, J. W. (2020). Improvements to target-based 3d LiDAR to camera calibration. *IEEE Access*, *8*, 134101–134110. URL: https://doi.org/10.1109/access.2020.3010734. doi:10.1109/access.2020.3010734.

Huang, L., & Barth, M. (2009). A novel multi-planar LIDAR and computer vision calibration procedure using 2d patterns for automated navigation. In *2009 IEEE Intelligent Vehicles Symposium*. IEEE. URL: https://doi.org/10.1109/ivs.2009.5164263. doi:10.1109/ivs.2009.5164263.

su Kim, E., & Park, S.-Y. (2019). Extrinsic calibration between camera and LiDAR sensors by matching multiple 3d planes. *Sensors*, *20*, 52. URL: https://doi.org/10.3390/s20010052. doi:10.3390/s20010052.

Liao, Y., Li, G., Ju, Z., Liu, H., & Jiang, D. (2017). Joint kinect and multiple external cameras simultaneous calibration. In *2017 2nd International Conference on Advanced Robotics and Mechatronics (ICARM)*. IEEE. URL: https://doi.org/10.1109/icarm.2017.8273179. doi:10.1109/icarm.2017.8273179.

Majumder, S., & Pratihar, D. K. (2018). Multi-sensors data fusion through fuzzy clustering and predictive tools. *Expert Systems with Applications*, *107*, 165 – 172. URL: http://www.sciencedirect.com/science/article/pii/S0957417418302677. doi:https://doi.org/10.1016/j.eswa.2018.04.026.

Melendez-Pastor, C., Ruiz-Gonzalez, R., & Gomez-Gil, J. (2017). A data fusion system of gnss data and on-vehicle sensors data for improving car positioning precision in urban environments. *Expert Systems with Applications*, *80*, 28 – 38. URL: http://www.sciencedirect.com/science/article/pii/S0957417417301641. doi:https://doi.org/10.1016/j.eswa.2017.03.018.

Mirzaei, F. M., Kottas, D. G., & Roumeliotis, S. I. (2012). 3d LIDAR–camera intrinsic and extrinsic calibration: Identifiability and analytical least-squares-based initialization. *The International Journal of Robotics Research*, *31*, 452–467. URL: https://doi.org/10.1177/0278364911435689. doi:10.1177/0278364911435689.

Oliveira, M., Castro, A., Madeira, T., Dias, P., & Santos, V. (2020a). A general approach to the extrinsic calibration of intelligent vehicles using ros. In M. F. Silva, J. Luís Lima, L. P. Reis, A. Sanfeliu, & D. Tardioli (Eds.), *Robot 2019: Fourth Iberian Robotics Conference* (pp. 203–215). Cham: Springer International Publishing.

Oliveira, M., Castro, A., Madeira, T., Pedrosa, E., Dias, P., & Santos, V. (2020b). A ROS framework for the extrinsic calibration of intelligent vehicles: A multi-sensor, multi-modal approach. *Robotics and Autonomous Systems*, *131*, 103558. URL: https://doi.org/10.1016/j.robot.2020.103558. doi:10.1016/j.robot.2020.103558.

Pandey, G., McBride, J., Savarese, S., & Eustice, R. (2010). Extrinsic calibration of a 3d laser scanner and an omnidirectional camera. *IFAC Proceedings Volumes*, *43*, 336–341. URL: https://doi.org/10.3182/20100906-3-it-2019.00059. doi:10.3182/20100906-3-it-2019.00059.

de Paula, M., Jung, C., & da Silveira, L. (2014). Automatic on-the-fly extrinsic camera calibration of onboard vehicular cameras. *Expert Systems with Applications*, *41*, 1997–2007. URL: https://doi.org/10.1016/j.eswa.2013.08.096. doi:10.1016/j.eswa.2013.08.096.

Penate-Sanchez, A., Andrade-Cetto, J., & Moreno-Noguer, F. (2013). Exhaustive linearization for robust camera pose and focal length estimation. *IEEE Transactions on Pattern Analysis*

and Machine Intelligence, *35*, 2387–2400. URL: https://doi.org/10.1109/tpami.2013.36. doi:10.1109/tpami.2013.36.

Pradeep, V., Konolige, K., & Berger, E. (2014). Calibrating a multi-arm multi-sensor robot: A bundle adjustment approach. In *Experimental Robotics* (pp. 211–225). Springer Berlin Heidelberg. URL: https://doi.org/10.1007/978-3-642-28572-1_15. doi:10.1007/978-3-642-28572-1_15.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., & Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA workshop on open source software* (p. 5). Kobe, Japan volume 3.

Rehder, J., Siegwart, R., & Furgale, P. (2016). A general approach to spatiotemporal calibration in multisensor systems. *IEEE Transactions on Robotics*, *32*, 383–398. URL: https://doi.org/10.1109/tro.2016.2529645. doi:10.1109/tro.2016.2529645.

Romero-Ramirez, F. J., Muñoz-Salinas, R., & Medina-Carnicer, R. (2018). Speeded up detection of squared fiducial markers. *Image and Vision Computing*, *76*, 38–47. URL: https://doi.org/10.1016/j.imavis.2018.05.004. doi:10.1016/j.imavis.2018.05.004.

dos Santos, F. N., Sobreira, H., Campos, D., Morais, R., Moreira, A. P., & Contente, O. (2016). Towards a reliable robot for steep slope vineyards monitoring. *Journal of Intelligent & Robotic Systems*, *83*, 429–444. URL: https://doi.org/10.1007/s10846-016-0340-5. doi:10.1007/s10846-016-0340-5.

Santos, L., Santos, F., Mendes, J., Costa, P., Lima, J., Reis, R., & Shinde, P. (2019). Path planning aware of robot's center of mass for steep slope vineyards. *Robotica*, *38*, 684–698. URL: https://doi.org/10.1017/s0263574719000961. doi:10.1017/s0263574719000961.

Sariff, N., & Buniyamin, N. (2006). An overview of autonomous mobile robot path planning algorithms. In *2006 4th Student Conference on Research and Development*. IEEE. URL: https://doi.org/10.1109/scored.2006.4339335. doi:10.1109/scored.2006.4339335.

Sturm, P. (2014). Pinhole camera model. In K. Ikeuchi (Ed.), *Computer Vision: A Reference Guide* (pp. 610–613). Boston, MA: Springer US. URL: https://doi.org/10.1007/978-0-387-31439-6_472. doi:10.1007/978-0-387-31439-6_472.

Verma, S., Berrio, J. S., Worrall, S., & Nebot, E. (2019). Automatic extrinsic calibration between a camera and a 3D lidar using 3D point and plane correspondences. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)* (pp. 3906–3912). Auckland, New Zealand: IEEE. doi:10.1109/ITSC.2019.8917108.

Wang, W., Sakurada, K., & Kawaguchi, N. (2017). Reflectance intensity assisted automatic and accurate extrinsic calibration of 3d LiDAR and panoramic camera using a printed chessboard. *Remote Sensing*, *9*, 851. URL: https://doi.org/10.3390/rs9080851. doi:10.3390/rs9080851.

Zhou, L., Li, Z., & Kaess, M. (2018). Automatic extrinsic calibration of a camera and a 3d LiDAR using line and plane correspondences. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. URL: https://doi.org/10.1109/iros.2018.8593660. doi:10.1109/iros.2018.8593660.

Zuniga-Noel, D., Ruiz-Sarmiento, J.-R., Gomez-Ojeda, R., & Gonzalez-Jimenez, J. (2019). Automatic multi-sensor extrinsic calibration for mobile robots. *IEEE Robotics and Automation Letters*, *4*, 2862–2869. URL: https://doi.org/10.1109/lra.2019.2922618. doi:10.1109/lra.2019.2922618.