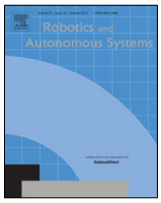
Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



Highlights

A ROS framework for the extrinsic calibration of intelligent vehicles: A multi-sensor, multi-modal approach

Robotics and Autonomous Systems xxx (xxxx) xxx

Miguel Oliveira, Afonso Castro*, Tiago Madeira, Eurico Pedrosa, Paulo Dias, Vítor Santos

- Intelligent vehicles require a package that calibrate all sensors at once, in order to operate consistently.
- The multimodal complete calibraton achieves similar accuracy when compared to standard pairwise calibrations.
- The multimodal calibraton approach works on ROS and uses several interactive tools that ease the calibraton procedure.
- The multimodal and multisensor calibraton does not change the initial robots transformation tree.

Please cite this article as: M. Oliveira, A. Castro, T. Madeira et al., A ROS framework for the extrinsic calibration of intelligent vehicles: A multi-sensor, multi-modal approach, Robotics and Autonomous Systems (2020) 103558, https://doi.org/10.1016/j.robot.2020.103558.

Robotics and Autonomous Systems xxx (xxxx) xxx

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

A ROS framework for the extrinsic calibration of intelligent vehicles: A multi-sensor, multi-modal approach

Miguel Oliveira [a,b], Afonso Castro [b,*], Tiago Madeira [a], Eurico Pedrosa [a], Paulo Dias [a,c], Vítor Santos [b]

[a] Institute of Electronics and Informatics Engineering of Aveiro, University of Aveiro, Portugal,
[b] Department of Mechanical Engineering, University of Aveiro, Portugal
[c] Department of Electronics, Telecommunications and Informatics, University of Aveiro, Portugal

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Extrinsic calibration
ROS
Optimization
Bundle adjustment
Intelligent vehicles
OpenCV

ABSTRACT

This paper proposes a general approach to the problem of extrinsic calibration of multiple sensors of varied modalities. This is of particular relevance for intelligent vehicles, which are complex systems that often encompass several sensors of different modalities. Our approach is seamlessly integrated with the *Robot Operating System* (ROS) framework, and allows for the interactive positioning of sensors and labelling of data, facilitating the calibration procedure. The calibration is formulated as a simultaneous optimization for all sensors, in which the objective function accounts for the various sensor modalities. Results show that the proposed procedure produces accurate calibrations, on par with state of the art approaches which operate only for pairwise setups.

1. Introduction

Intelligent vehicles require a considerable amount of on-board sensors, often of multiple modalities (e.g. camera, LiDAR, etc.) in order to operate consistently. The combination of the data collected by these sensors requires a transformation or projection of data from one sensor coordinate frame to another. The process of estimating these transformations between sensor coordinate systems is called extrinsic calibration. An extrinsic calibration between two sensors requires an association of data from one sensor to the data of another. By knowing these data associations, an optimization procedure can be formulated to estimate the parameters of the transformation between those sensors that minimizes the distance between associations. Most calibration approaches make use of calibration patterns, i.e., objects that are robustly and accurately detected by distinct sensor modalities.

Although there have been many solutions available in the literature, on the topic of calibration, there is no straightforward solution for the calibration of multiple sensors in intelligent vehicles, or robots in general. There are multiple factors that contribute to this lack of solutions. The majority of works on calibration focus on sensor to sensor pairwise calibrations: between only cameras [1–5] or between cameras and LiDARs [6–10]. When considering pairwise combinations of sensors, there are several possibilities, according to the modality of each of the sensors in the pair. Most of them have been addressed in the literature: RGB to RGB camera calibration [1–5,7]; RGB to depth camera (RGB-D cameras) calibration [11–16]; camera to 2D LiDAR [6,7,10,15,17–20]; 2D LiDAR to 3D LiDAR [8]; camera to 3D LiDAR [10,21,22]; and camera to radar [23].

Nonetheless, all these approaches have the obvious shortcoming of operating only with a single pair of sensors, which is not directly applicable to the case of intelligent vehicles, or more complex robotic systems in general. To be applicable in those cases, pairwise approaches must be arranged in a graph-like sequential procedure, in which one sensor calibrates with another, that then relates to a third sensor, and so forth. Another option is to establish one sensor as the reference sensor and link all other sensors to it. In this case, the graph of transformations between sensors results in a one level pyramid, which contains the reference sensor on top and all other sensors at the base. One example is [7], in which a methodology for calibrating the ATLASCAR2 autonomous vehicle [24] is proposed, wherein all sensors are paired with a reference sensor. Sequential pairwise approaches have three major shortcomings: (*i*) transformations are estimated using only data provided from the selected sensor tandem, despite the fact that data from additional sensors could be available and prove relevant to the overall accuracy of the calibration procedure; (*ii*) sensitivity to cumulative errors, since the transformations are computed in a sequence (in fact, this does not occur in [7], since the pose of a specific sensor only depends

* Corresponding author.
E-mail addresses: mriem@ua.pt (M. Oliveira), afonsocastro@ua.pt (A. Castro), tiagomadeira@ua.pt (T. Madeira), efp@ua.pt (E. Pedrosa), paulo.dias@ua.pt (P. Dias), vitor@ua.pt (V. Santos).

https://doi.org/10.1016/j.robot.2020.103558
0921-8890/

on the reference sensor pose); (*iii*) structure of transformation graph is enforced by the nature of the calibration procedure, rather than being defined by the preference of the programmer, which could compromise some robot functionalities. Fig. 1 shows a conceptual example in which these problems are visible.

There are a few works which address the problem of calibration from a multi-sensor, simultaneous optimization, perspective. In [25], a joint objective function is proposed to simultaneously calibrate three RGB cameras with respect to an RGB-D camera. Authors report a significant improvement in the accuracy of the calibration. In [26], an approach for joint estimation of both temporal offsets and spatial transformations between sensors is presented. 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. Moreover, the approach does not require the usage of calibration patterns for the LiDARs, using the planes present in the scene for that purpose. In [27], a joint calibration of the joint offsets and the sensors locations for a PR2 robot is proposed. This method takes sensor uncertainty into account and is modelled in a similar way to the bundle adjustment problem.

Our approach is similar to [27], in the sense that we also employ a bundle adjustment-like optimization procedure. However, our approach is not focused on a single robotic platform, rather it is a general approach that is applicable to any robotic system, which also relates it with [26].

This paper is an extension of [28], where the general approach was originally proposed. This extension focuses on the comparison of this work against state of the art calibration approaches, i.e. methodologies provided by *Open Source Computer Vision Library* (OpenCV) [29] as well as the calibration method from [26].

Robot Operating System ROS [30] based architectures are the standard when developing robots. There are several ROS based calibration packages available.[1][2][3] In addition, some approaches are well integrated with ROS since the input data for the calibration is provided as a *rosbag* file. Despite this, no approach provides a complete solution for the calibration of intelligent vehicles. Thus, the seamless integration with ROS became a core component of the proposed approach. To that end, the proposed calibration procedure is self-configured using the standard ROS robot description files, the Unified Robot Description Format (URDF), and provide several tools for sensor positioning and data labelling based on RVIZ interactive markers.

The remainder of this paper is organized as follows: Section 2 describes the methodologies used to set up an optimization procedure which calibrates the system. In this section, several auxiliary tools for labelling data and positioning sensors are described; Section 3 details the optimization procedure and how it is cast as a bundle adjustment problem; Section 4 provides comparisons with established OpenCV calibration methodologies; finally, Section 5 provides conclusions and future work.

2. ROS based calibration setup

A schematic of the proposed calibration procedure is displayed in Fig. 2. It consists of five components: configuration; interactive positioning of sensors; interactive labelling of data; collection of data; and finally, the optimization procedure. Each component will be described in detail in the following subsections.

2.1. Configuration of the calibration procedure

Robotic platforms are described in ROS using a *xml* file called URDF. We propose to extend the URDF description files of a robot in order to provide information necessary for configuring how the calibration should be carried out. A new URDF element, named *calibration*, is introduced specifically for the purpose of calibrating. Each *calibration element* describes a sensor to be calibrated. The element contains information about the calibration parent and child links, which define the partial transformation that is optimized.

2.2. Interactive positioning of sensors

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. Thus, it is expected that, by ensuring an accurate first guess for the sensor poses, there is less likelihood of falling into local minima. We propose to solve this problem in an interactive fashion: the system parses the URDF robot description and creates an rviz interactive marker associated with each sensor. It is then possible to move and rotate the interactive markers. This provides a simple, interactive method to manually calibrate the system or, alternatively, to easily generate plausible first guesses for the poses of the sensors. Real time visual feedback is provided by the observation of the bodies of the robot model (e.g. where a LiDAR is placed w.r.t. the vehicle), and also by the data measured by the sensors (e.g. how well the measurements from two LiDARs match). An example of this procedure can be watched at https://youtu.be/zyQF7Goclro.

2.3. Interactive data labelling

Since the goal is to propose a calibration procedure that operates on multi-modal data, a calibration pattern adequate to all available sensor modalities must be selected. A chessboard pattern is a common calibration pattern, in particular for RGB and RGB-D cameras. To label image data, one of the many available image-based chessboards detectors is used (*Find Chessboard Corners* OpenCV function[4]). In the case of 2D LiDAR data, it is not possible to robustly detect the chessboard, since there are often multiple planes in the scene derived from other structures, such as walls and doors. To solve this, we propose an interactive approach which requires minimal user intervention: rviz interactive markers are positioned along the LiDAR measurement planes and the user drags the marker to indicate where in the data the chessboard is observed. This is done by clustering the LiDAR data, and selecting the cluster which is closer to the marker. This interactive procedure is done only once, since it is then possible to track the chessboard robustly. Fig. 3 shows an example of the labelling of 2D LiDAR data. This interactive data labelling procedure is showcased in https://youtu.be/9pGXShLIEHw.

2.4. Collecting data

Usually, different sensors stream data at different frequencies. However, to compute the associations between the data of multiple sensors, temporal synchronization is required. While some approaches require hardware synchronization to operate [26], in the current method this 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

[1] http://wiki.ros.org/camera_calibration/Tutorials/StereoCalibration.
[2] http://wiki.ros.org/openni_launch/Tutorials/IntrinsicCalibration.
[3] https://github.com/code-iai/iai_kinect2.
[4] https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#findchessboardcorners.

Please cite this article as: M. Oliveira, A. Castro, T. Madeira et al., A ROS framework for the extrinsic calibration of intelligent vehicles: A multi-sensor, multi-modal approach, Robotics and Autonomous Systems (2020) 103558, https://doi.org/10.1016/j.robot.2020.103558.

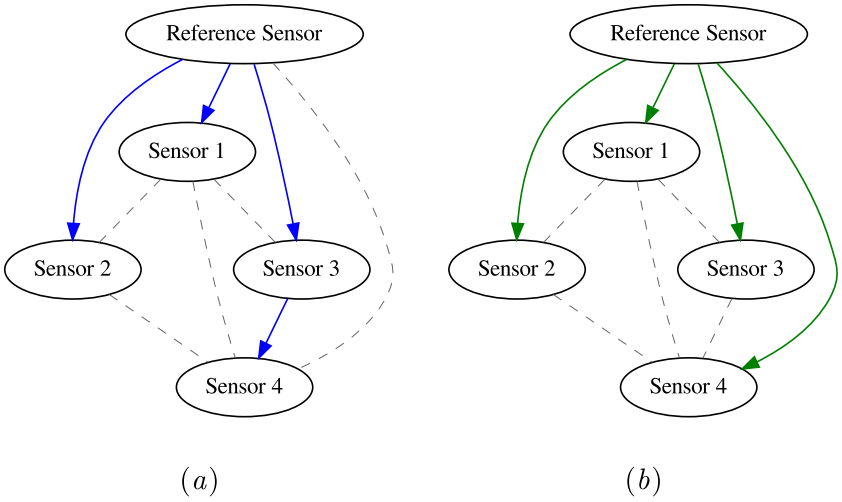


Fig. 1. Two methodologies for solving the calibration of complex systems using pair-wise approaches: (*a*) sequential pairwise; (*b*) one level pyramid using a reference sensor. The estimated transformations use the arrangements shown in solid colour arrows. Other possible arrangements are presented in dashed grey lines. Note that in both cases only a subset of the available transformations is used.

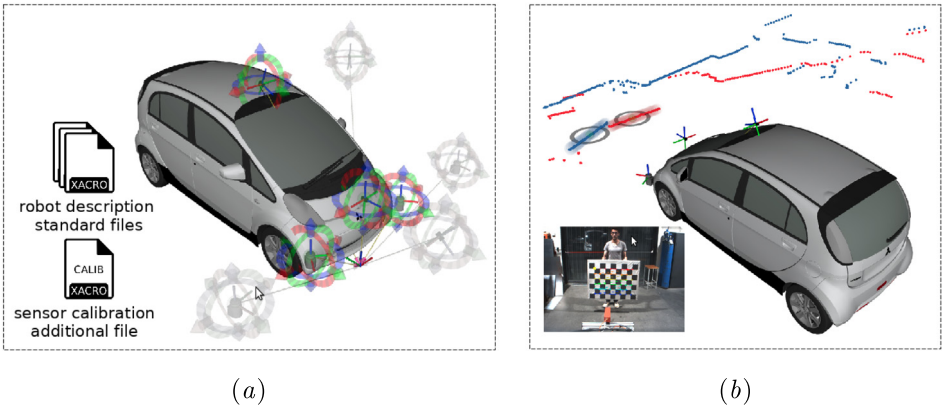


Fig. 2. The proposed calibration procedure: (*a*) initialization from *xacro* files and interactive first guess; (*b*) data labelling and collecting.

scenes, the problem of data desynchronization is not observable, which warrants the assumption that for each captured collection the sensor data is synchronized. We refer to these snapshot recordings of multi-sensor data as data collections.

This information is stored in a JSON file that will be read by the optimization procedure afterwards. The JSON file contains abstract information about the sensors, such as the sensor transformation chain, among others, and specific information about each collection, i.e., sensor data, partial transformations, and data labels. It is important to note 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 chessboard so that the calibration becomes more reliable. This concern is common to the majority of calibration procedures.

2.5. Sensor poses from partial transformations

The representation of a complex, multi-sensor system requires the creation of a transformation graph. For this purpose, ROS uses a graph tree referred to as *tf tree* [31]. One critical factor for any calibration procedure is that it should not change the structure of that existing *tf tree*. The reason for this is that the *tf tree*, derived from the URDF files by the robot state publisher,[5] also supports additional functionalities, such as robot visualization or collision detection. If the *tf tree* changes due to the calibration, those functionalities may be compromised or require some redesigning. To accomplish this, we propose to compute the pose of any particular sensor (i.e., the transformation from the Reference Link, also known as World, to that Sensor) as an aggregate transformation **A**, obtained after the chain of transformations for that particular sensor, extracted from the topology of the *tf tree*:

$$\mathbf{A} = \prod_{i=\text{world}}^{\text{sensor}-1} {}^{i}\mathbf{T}_{i+1} = \overbrace{\prod_{i=\text{world}}^{\text{parent}-1} {}^{i}\mathbf{T}_{i+1}}^{\text{prior links}} \cdot \overbrace{{}^{\text{parent}}\mathbf{T}_{\text{child}}}^{\text{to be calibrated}} \cdot \overbrace{\prod_{i=\text{child}}^{\text{sensor}-1} {}^{i}\mathbf{T}_{i+1}}^{\text{later links}}, \tag{1}$$

where ${}^{i}\mathbf{T}_{i+1}$ represents the partial transformation from the *i*th to the $i+1$ link, and *parent* and *child* are the indexes of the *calibration parent* and *calibration child* links in the sensor chain, respectively.

Our approach preserves the predefined structure of the *tf* tree, since, during optimization, only one partial transformation contained in the chain is altered (the one in blue in Eq. (1)). This computation is performed within the optimization's cost function. Therefore, a change in one partial transformation affects the global sensor pose, and consequently, the error to minimize. The optimization may target multiple links of each chain, and is

[5] http://wiki.ros.org/robot_state_publisher.

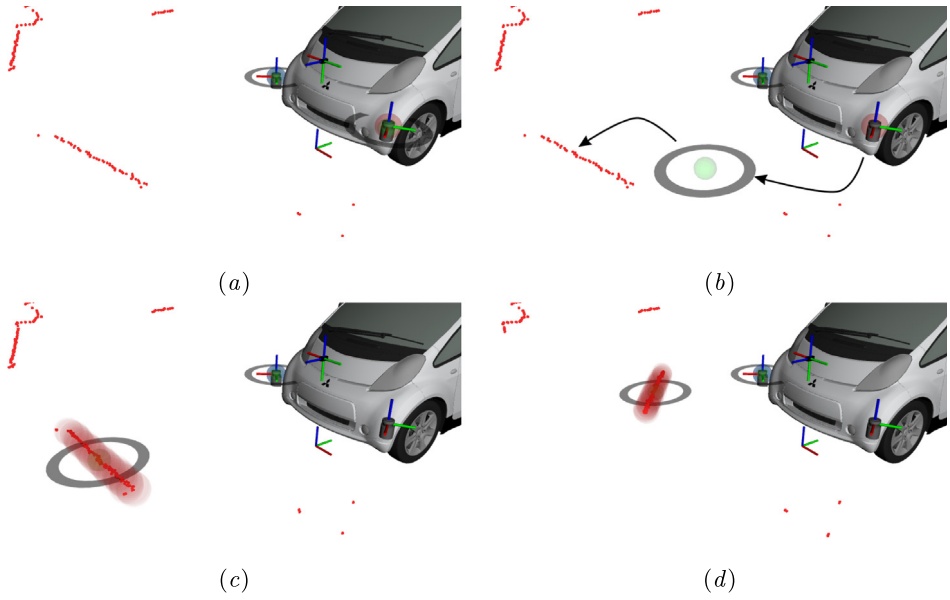


Fig. 3. Interactive labelling of 2D LiDAR data: (*a*) creation of interactive marker on the sensor body, (*b*) dragging and dropping the marker on top of the data cluster containing the chessboard plane, (*c*) and (*d*) subsequent automatic tracking of the chessboard plane.

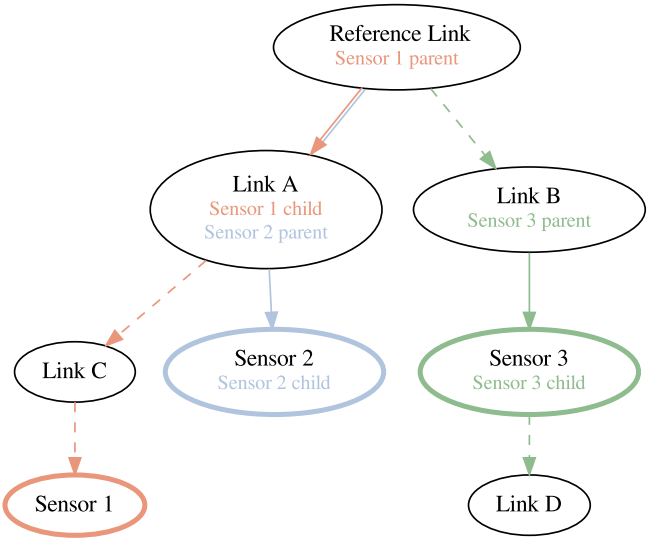


Fig. 4. Conceptual transformation graph for a complex robotic system. Each sensor has a respective calibration partial transformation, denoted by the solid edges. Dashed edges contain transformations which are not optimized (they may be static or dynamic). Each sensor has a corresponding link to which the data it collects is attached, denoted in the figure by the solid thin ellipses. Very few approaches in the literature are capable of calibrating such a system while preserving the initial structure of the graph of transformations.

agnostic to whether the remaining links are static or dynamic, since all existing partial transformations are stored for each data collection.

To the best of our knowledge, our approach is one of few which maintains the structure of the transformation graph before and after optimization. This is a feature that is often overlooked, yet it is of critical practical importance for the selection of a calibration framework.

Taking the example of Fig. 4, consider that *Sensor 1* is mounted on top of a pan and tilt unit, where ${}^{LinkA}\mathbf{T}_{Link\ C}$ corresponds to the pan movement, and ${}^{LinkC}\mathbf{T}_{Sensor\ 1}$ represents the tilt motion. For this particular case, Eq. (1) becomes:

$$\mathbf{A}^{Sensor\ 1} = \overbrace{\mathbf{I}}^{\text{prior links}} \cdot \overbrace{{}^{R\ Link}\mathbf{T}_{Link\ A}}^{\text{to be calibrated}} \cdot \overbrace{\underbrace{{}^{Link\ A}\mathbf{T}_{Link\ C}}_{\text{pan motion}} \cdot \underbrace{{}^{Link\ C}\mathbf{T}_{Sensor\ 1}}_{\text{tilt motion}}}^{\text{later links}}, \tag{2}$$

where $\mathbf{I}$ is the identity matrix (since there are no prior links), and the pan and tilt motions are coloured in red to denote that these transformations are dynamic and, as a consequence, may also change from collection to collection.

Another example is the one of *Sensor 2*: it contains an aggregate transformation that also includes the partial transformation optimized w.r.t. *Sensor 1*, resulting in the following aggregate transformation:

$$\mathbf{A}^{Sensor\ 2} = \overbrace{\underbrace{{}^{R\ Link}\mathbf{T}_{Link\ A}}_{\text{to be calibrated}}}^{\text{prior links}} \cdot \overbrace{{}^{Link\ A}\mathbf{T}_{Sensor\ 2}}^{\text{to be calibrated}} \cdot \overbrace{\mathbf{I}}^{\text{later links}}, \tag{3}$$

which is also directly derived from Eq. (1).

Such complex arrangements are seldom tackled by a single calibration approach, even less in a transparent way by the same general formalism. The proposed optimization of partial transformations achieves this goal. We consider the ability to preserve the structure of the *tf tree* as a key feature of the proposed framework: from a practical standpoint, since it facilitates the integration into ROS, both before and after the optimization; and, moreover, from a conceptual perspective, since this formulation is general and adequate to handle most calibration scenarios.

3. Calibration procedure

The goal of general optimization procedures is to find the parameter configuration that results in the smallest function value. This function, which depends on the optimization parameters Φ is known as the *objective function*. For the purpose of calibrating the multi-modal sensors of a robotic platform, like the ATLASCAR2 intelligent vehicle, the objective of this optimization is to estimate the pose of each sensor relatively to a reference link (*base link* for ATLASCAR2 case).

Please cite this article as: M. Oliveira, A. Castro, T. Madeira et al., A ROS framework for the extrinsic calibration of intelligent vehicles: A multi-sensor, multi-modal approach, Robotics and Autonomous Systems (2020) 103558, https://doi.org/10.1016/j.robot.2020.103558.

3.1. Optimization parameters

An extrinsic calibration translates into a pose estimation. Thus, the set of parameters to optimize, defined as Φ, must contain parameters that together define the pose of each sensor. As discussed in the beginning of Section 2, we propose to maintain the initial structure of the transformation graph, and thus only optimize one partial transformation per sensor. In the example of Fig. 4, these partial transformations are denoted by solid arrows. Since the usage of camera sensors is considered, it is also possible to introduce the intrinsic parameters of each camera in the set Φ. Our goal is to define an objective function that is able to characterize sensors of different modalities.

Pairwise methodology for devising the cost function results in complex graphs of exhaustive definition of relationships. For every existing pair of sensors, these relationships must be established according to the modality of each of the sensors, and, although most cases have been addressed in literature, as discussed in Section 1, a problem of scalability remains inherent to such a solution. To address this issue, we propose to structure the cost function in a sensor to calibration pattern paradigm, similar to what is done in bundle adjustment. That is, the positions of 3D points in the scene are jointly refined with the poses of the sensors. These 3D points correspond to the corners of the calibration chessboard. What is optimized is actually the transformation that takes these corners from the frame of reference of the chessboard to the world, for every collection. All variables must have some initial value, so that the optimizer may compute the first error, and start to refine the values in order to obtain the minimum of the cost function. The first guess for each chessboard is obtained by computing the pose of a chessboard detection in one of the cameras available. The output is a transformation from the chessboard reference frame to the camera's reference frame. Since we already have the first guess for the poses of each sensor, calculated as an aggregate transformation A (see Eq. (1)), to obtain the transformation from the chessboard reference frame to the world (an external and absolute frame), the following calculation is applied:

$$ {}^{\text{chess}}\mathbf{T}_{\text{world}} = \overbrace{{}^{\text{camera}}\mathbf{A}_{\text{world}}}^{\text{Eq. (1)}} \cdot \overbrace{{}^{\text{chess}}\mathbf{T}_{\text{camera}}}^{\text{chess detection}}, \tag{4} $$

where chess and camera refer to chessboard and camera coordinate frames, respectively. Thus, the set of parameters to be optimized Φ, contains the transformation represented in Eq. (4), for each collection, along with the poses of each sensor:

$$ \Phi = \Big[\overbrace{\mathbf{x}_{m=1}, \mathbf{r}_{m=1}, \mathbf{i}_{m=1}, \mathbf{d}_{m=1}, \ldots, \mathbf{x}_{m=M}, \mathbf{r}_{m=M}, \mathbf{i}_{m=M}, \mathbf{d}_{m=M}}^{\text{Cameras}}, \overbrace{\mathbf{x}_{n=1}, \mathbf{r}_{n=1}, \ldots, \mathbf{x}_{n=N}, \mathbf{r}_{n=N}}^{\text{LiDARs}}, \overbrace{\ldots}^{\text{Other modalities}}, \overbrace{\mathbf{x}_{k=1}, \mathbf{r}_{k=1}, \ldots, \mathbf{x}_{k=K}, \mathbf{r}_{k=K}}^{\text{Calibration object}} \Big] \tag{5} $$

where m refers to the mth camera, of the set of M cameras, n refers to the nth LiDAR, of the set of N LiDARs, k refers to the chessboard detection of the kth collection, contained in the set of K collections, $\mathbf{x}$ is a translation vector $[t_x, t_y, t_z]$, $\mathbf{r}$ is a rotation represented through the axis/angle parameterization $[r_1, r_2, r_3]$ (where the vector $[r_1, r_2, r_3]$ is used to represent the axis and its norm the angle), $\mathbf{i}$ is a vector of a camera's intrinsic parameters $[f_x, f_y, c_x, c_y]$, and $\mathbf{d}$ is a vector of camera's distortion coefficients $[d_0, d_1, d_2, d_3, d_4]$. The initial estimate for the intrinsic parameters is obtained using any intrinsic camera calibration tool. The axis/angle parameterization was chosen because it has 3 components and 3 degrees of freedom, making it a fair parameterization, since it does not introduce more numerical sensitivity than the one inherent to the problem itself [32].

At this point, there are six parameters per sensor, related to the pose of each one, to be enhanced. These values compose the geometric transformation that will be calibrated. The cost function will compute the residuals based on an error (in pixels for RGB cameras and in millimetres for LiDARs) between the re-projected position of the chessboard, estimated by all transformations, and the position of the calibration pattern detected by each sensor.

3.2. Objective function

The cost function for this optimization, $F(\Phi)$, can be thought of as the sum of several sub-functions that compose a vector function, where, for every modality of sensor added to the calibration, a new sub-function is defined accordingly, which allows for the minimization of the error associated with the pose of sensors of that modality. Thus, the optimization procedure can be defined as:

$$ \arg\min_{\Phi} F(\Phi) = \frac{1}{2} \sum_{i} \left\| f_i(\Phi_{i_1}, \ldots, \Phi_{i_k}) \right\|^2 \tag{6} $$

where $f_i(\cdot)$ is the objective sub-function for the ith sensor with the respective parameters block $\{\Phi_{i_1}, \ldots, \Phi_{i_k}\}$, being k the parameters number of each objective sub-function. In other words, the scalar cost function of this optimization is the sum of the squares of the returned values from a vector function, divided by two. Each different sensor has an inherent sub-function, that depends on the sensor modality. The value of all these sub-functions is a vector with the errors (residuals) associated to the re-projection of the calibration pattern points. Since for the ATLASCAR2 intelligent vehicle we are considering four sensors (two cameras and two 2D LiDARs), the objective function is composed by the vector values of four sub-functions, two of each type. Each different sub-function is detailed in the next sub-sections.

3.2.1. Camera sub-function

When the sensors are cameras, their calibration is performed as a bundle adjustment [33], and as such, the sub-function created is based on the average geometric error corresponding to the image distance between a projected point and a detected one. The 3D points corresponding to the corners of the calibration chessboard are captured by one or more cameras in each collection. Each camera is defined by its pose relative to a reference link and intrinsic parameters. After the desired acquisitions are completed, the 3D points are projected from the world into the images and the 2D coordinates are compared to the ones obtained by detection of the calibration pattern in the corresponding images. The positions of the 3D points in the world are obtained by applying the transformation described in Eq. (4) to the chessboard corner points defined in the chessboard detection's reference frame. The goal of this cost sub-function is to adjust the initial estimation of the camera parameters and the position of the points, in order to minimize the average reprojection error f_{camera}, given by:

$$ f_{\text{camera}} = \begin{bmatrix} \ell^2(\mathbf{x}_{c=1}, \hat{\mathbf{x}}_{c=1}) & \ell^2(\mathbf{x}_{c=2}, \hat{\mathbf{x}}_{c=2}) & \cdots & \ell^2(\mathbf{x}_{c=C}, \hat{\mathbf{x}}_{c=C}) \end{bmatrix}^{\mathsf{T}} \tag{7} $$

where ℓ^2 is the Euclidean distance between two vectors, c denotes the index of the chessboard corners, $\mathbf{x}_c$ denotes the pixels coordinates of the measured points (given by chessboard detection), and $\hat{\mathbf{x}}_c$ are the projected points, given by the relationship between a 3D point in the world and its projection on an image plane.

Please cite this article as: M. Oliveira, A. Castro, T. Madeira et al., A ROS framework for the extrinsic calibration of intelligent vehicles: A multi-sensor, multi-modal approach, Robotics and Autonomous Systems (2020) 103558, https://doi.org/10.1016/j.robot.2020.103558.

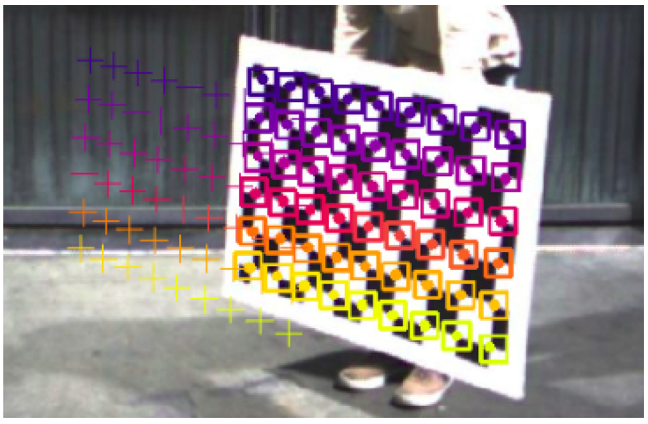


Fig. 5. Example of reprojection of chessboard corners during the optimization procedure: squares denote the position of the detected chessboard corners (*ground truth* points); crosses denote the initial position of each projected corner; points denote the current position of the projected corners.

By knowing the real size of the chessboard squares, the 3D coordinates of all corners relatively to the chess frame can be inferred. Note that the z value will be, for every point, zero, since the chessboard is in the XoY plane. After obtaining the 3D coordinates of all corners in reference to the chessboard frame, the objective function computes the coordinates of the points relatively to the camera link through multiplying by the geometric transformation between the *base link* (reference frame for the ATLASCAR2 example) and the calibration pattern frame and by the transform between the camera link and the *base link*:

$$\mathbf{p}^{camera} = {}^{\text{camera}}\mathbf{T}_{\text{world}} \cdot {}^{\text{world}}\mathbf{T}_{\text{chess}} \cdot \mathbf{p}^{chess} \tag{8}$$

where $\mathbf{p}^{chess}$ refers to the x, y, z coordinates of a chessboard corner, defined in the local chessboard coordinate frame, and $\mathbf{p}^{camera}$ refers to the x, y, z coordinates of the same chessboard corner, defined in the camera link. In fact, both $\mathbf{p}^{chess}$ and $\mathbf{p}^{camera}$ are the homogenized matrices of the coordinates so that Eq. (8) is mathematically correct.

Note that the parameters to be optimized define the chessboard to world transformation, and that the world to camera transformation is computed from an aggregate of several partial transformations, one of which is defined by other parameters being optimized; furthermore, the intrinsic matrix is dependent on parameters which are accounted for in the optimization. As is expected, the re-projected points become closer to the *ground truth* corners during the optimization procedure. Fig. 5 shows the difference between the initial position of the chessboard corners, projected from the 3D world to the camera image, 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. 5) almost perfectly match the *ground truth* points (squares in Fig. 5).

3.2.2. Laser sub-function

Finally, for the case of 2D LiDARs, the sub-function only considers the two border points, among all the measurements that are related to the chessboard plane, to compute the error associated to the pose of the LiDAR and the chessboard. In order to calculate the residuals that this cost sub-function should return, the detected points' 3D coordinates from the chessboard frame are required. During the calibration setup stage, when the information of a time stamp is saved, the ranges of all measurements that the LiDAR is detecting are stored, as well as the information about this same LiDAR and the indexes of the ranges that correspond to the plane where the chessboard is. With the optimization parameters of the chessboard pose and the LiDAR pose (computed accordingly to Eq. (1)), both relative to the *base link*, the 3D coordinates of each labelled measurement of the point cloud in the chessboard frame are known:

(a) (b)

Fig. 6. Chessboard: graphics visualization of the created grid and boundary correspondent to the real chessboard (a); image of the real chessboard (b).

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}^{\text{chess}} = {}^{\text{chess}}\mathbf{T}_{\text{world}} \cdot {}^{\text{world}}\mathbf{T}_{\text{lidar}} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}^{\text{lidar}}. \tag{9}$$

Finally, with the coordinates from the chessboard frame, of both the first and the last points of the cluster extracted in the labelling stage, it is possible to compute the error evaluated by this cost sub-function. The error is based on the distance between each one of the limit points (the first and the last index) of the selected ranges and the chessboard surface boundaries. There are two computed distances for each point: orthogonal and longitudinal. The orthogonal distance is the z absolute value of the coordinates, in the calibration pattern frame, of the LiDAR data measurement. In an ideal setting, the z value should be zero, since the chessboard plane is on the XoY plane. This is why any value different from zero means that the optimization parameters (sensor pose and chess pose) are not yet correct. The longitudinal distance is the Euclidean distance between the x and y coordinates, in the calibration pattern frame, of the LiDAR data measurement and the x and y coordinates of the closest point that belong to the limit of the physical board that is being detected. In order to compute this distance, it is essential to create a group of points that represent the boundaries of the chessboard. By knowing the size of the board, the size of each chess square, and that the chess frame origin matches with the first (top left) chess corner, the coordinates were calculated and the points of the board boundaries were manually defined. The size of the border between the chess corner grid and the end of the physical chessboard had to be measured so that this step could be implemented. In Fig. 6, we can see the grid of the chess corners and a line around it: that line marks the limit of the board. This solid line has some points within it, which are going to be compared to the LiDAR data measured ones.

Again, the optimizer will search for the closest limit point to each one of the studied LiDAR data measurement coordinates and then compute the longitudinal distance. Thus, the LiDAR sub-function f_{lidar} is defined as:

$$f_{\text{lidar}} = \left[|z_1^{\text{chess}}| \quad \ell^2(\mathbf{p}_1^{boardlimit}, \mathbf{p}_1^{chess}) \quad |z_2^{\text{chess}}| \quad \ell^2(\mathbf{p}_2^{boardlimit}, \mathbf{p}_2^{chess}) \right]^{\top} \tag{10}$$

where

$$\mathbf{p}^{boardlimit} = \begin{bmatrix} x \\ y \end{bmatrix}^{\text{boardlimit}}, \tag{11}$$

$$\mathbf{p}^{chess} = \begin{bmatrix} x \\ y \end{bmatrix}^{\text{chess}}, \tag{12}$$

Fig. 7. ATLASCAR2: Autonomous vehicle from the Department of Mechanical Engineering of the University of Aveiro; the sensors are indicated by the Ellipses.

and z^{chess} is the third coordinate value of the range measurement points transformed to the chessboard's coordinate frame.

3.3. Sensors pose calibration: Optimization

The cost function $F(\Phi)$ from Eq. (6) is minimized using a least-squares approach.[6] Least-squares finds a local minimum of a scalar cost function, with bounds on the variables, by having an m-dimensional real residual function of n real variables. As such, we choose this minimization approach as it is the best fit for our problem.

4. Results

To assess the performance of the proposed calibration approach, we used an intelligent vehicle as test bed. The ATLASCAR2 [24] is an electric vehicle (Mitsubishi i-MiEV) with several sensors onboard. In this work four sensors were considered: two 2D LiDARs and two RGB cameras. Thus, two different modalities of sensors are used. The sensors are designated as follows: *left laser*, *right laser*, *top left camera* and *top right camera*. Fig. 7 shows the ATLASCAR2 vehicle.

The proposed approach is used to calibrate the four selected sensors simultaneously. Nonetheless, as discussed above, there are no approaches which provide an off-the-shelf multi-sensor multi-modal calibration. As such, in order to evaluate this approach, we provide comparisons against other pairwise methodologies, which are abundant in the field, as was mentioned in Section 1. Note that, in the following comparisons, the results given by the proposed approach for a particular pair of sensors are obtained using a complete system calibration. On the other hand, the alternative methodologies calibrate a single pair of sensors. In this sense, the comparison methodology is not favourable to the proposed approach, since the other approaches are specialized in the case being evaluated.

In the following lines, two tests are detailed: the first is a camera-to-camera evaluation which compares several calibration methods in a pairwise fashion, while the second characterizes the proposed joint optimization over time providing global metrics.

[6] 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.

4.1. Camera to camera

The methodology used to compute the error of the calibrated poses of the *top right camera* and the *top left camera* is based on the distance between pixel coordinates. These coordinates are, on the one hand, the detected chessboard corners (*ground truth*) of the *top right camera* and, on other hand, the coordinates of the projections of those corners, to the *top left camera*, using the transformation between the cameras which is the output of the calibration.

To transform pixels from one camera to the other, we start from the projection of the 3D world coordinates to the image of a camera:

$$\mathbf{p} = \mathbf{K} \cdot \left[\mathbf{R} \mid \mathbf{t}\right] \cdot \mathbf{P} \tag{13}$$

where $\mathbf{P}$ refers to the 3D homogeneous coordinates of the corners as viewed in the chessboard frame; $\mathbf{p}$ is a vector composed by the u, v and w values, in which: $x_{\text{pixel}} = u/w$ and $y_{\text{pixel}} = v/w$, allowing for the direct extraction of image coordinates from this vector; $\left[\mathbf{R} \mid \mathbf{t}\right]$ is the non-homogeneous geometric transformation matrix from the camera frame to the chessboard frame, $\mathbf{K}$ represents the camera's intrinsic matrix. Eq. (13) can be applied to each camera separately. Since the 3D chessboard corner coordinates are defined in the chessboard frame, the value of Z will be 0 for all corners, because they all lie on the XoY plane: $Z^{\text{chess}} = 0$. As a result, Eq. (13) may be simplified as follows:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}^{\text{chess corners}} \tag{14}$$

which is equivalent to:

$$\mathbf{p}^{\text{camera}} = \mathbf{K} \cdot {}^{\text{camera}}\mathbf{T}'_{\text{chess}} \cdot \mathbf{P}^{\text{chess}}, \tag{15}$$

where the geometric transformation matrix ${}^{\text{camera}}\mathbf{T}'_{\text{chess}}$ is a portion of the ${}^{\text{camera}}\mathbf{T}_{\text{chess}}$ matrix, as detailed in Eq. (15). We use (15) for both cameras, and relate both expressions by the 3D coordinates of the chessboard corners (which are the same for both cameras), resulting in:

$$\mathbf{p}^{\text{cam2}} = \mathbf{K}^{\text{cam2}} \cdot {}^{\text{cam2}}\mathbf{T}'_{\text{chess}} \cdot \left({}^{\text{cam1}}\mathbf{T}'_{\text{chess}}\right)^{-1} \cdot \left(\mathbf{K}^{\text{cam1}}\right)^{-1} \cdot \mathbf{p}^{\text{cam1}}, \tag{16}$$

where cam1 and cam2 refer to the *top left* and *top right* cameras, respectively. This formulation provides the relation between image coordinates of the chessboard corners for both camera images of each collection. Notice, however, that calibration methods output the transformation between sensors, in this case between cameras, while Eq. (16) requires transformations from the cameras to the chessboard.

Some approaches, as for example the proposed approach, also estimate the pose of the chessboards (see parameters of the calibration objects in Eq. (5)). Thus, at first glance, one could think of using these transformations directly in Eq. (16). However, these chessboard poses are estimated for a given training dataset, and cannot be accurately used for other datasets. Moreover, as said before, not all calibration approaches output the pose of the chessboards (e.g. OpenCV stereo calibrate). Instead, calibration approaches provide the transformation between cameras. By arbitrarily selecting one camera from which the chessboard pose is determined through the `solvePNP` function (we have used cam1, but tests have shown that the alternative provided similar results) and using the transformation ${}^{\text{cam1}}\mathbf{T}_{\text{cam2}}$ estimated by the calibration approaches, it is possible to determine the transformation of the other camera to the chess, as follows:

$${}^{\text{cam2}}\mathbf{T}_{\text{chess}} = \overbrace{\left({}^{\text{cam1}}\mathbf{T}_{\text{cam2}}\right)^{-1}}^{\text{calibration}} \cdot \overbrace{{}^{\text{cam1}}\mathbf{T}_{\text{chess}}}^{\texttt{solvePnP}}. \tag{17}$$

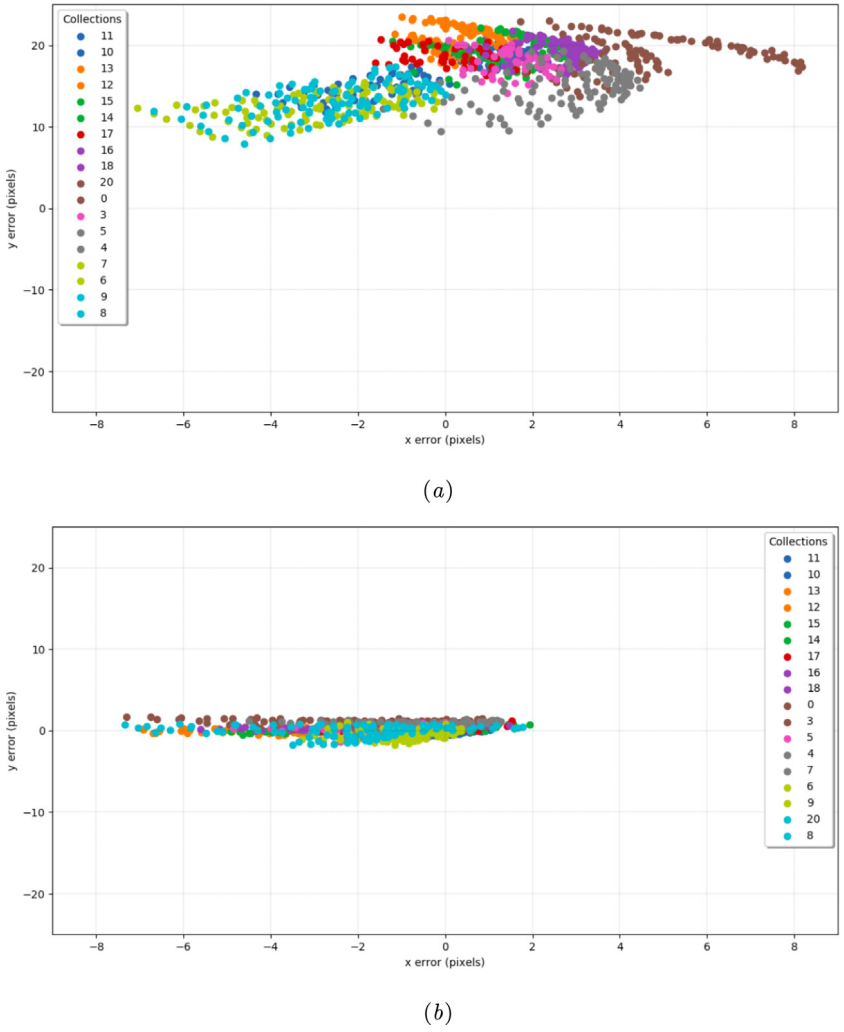


Fig. 8. Pixel coordinates errors between projected (expected) chessboard corners and the *ground truth* indexes, from *top left camera* to *top right camera*, for each collection, before the optimization procedure (*a*) and after the optimization procedure (*b*).

Table 1
Average errors and standard deviations along both directions, before and after the optimization. Values in pixels.

Values	Average error		Standard deviation	
	Initial	Final	Initial	Final
x error	2.25	1.64	2.68	1.69
y error	17.09	0.53	3.32	0.62
Both	17.34	1.83	8.71	1.51

From this expression the partial matrices ${}^{\text{cam1}}\mathbf{T}'_{\text{chess}}$ and ${}^{\text{cam2}}\mathbf{T}'_{\text{chess}}$ are derived. Then, we apply Eq. (16) to compute the corner coordinates on the *top right camera* image, as projected from the detection of the *top left camera* image. The error is computed by measuring the difference between expected and projected corner coordinates on the *top right camera* image:

$$\begin{bmatrix} \text{x error} \\ \text{y error} \end{bmatrix}^{\text{top right camera}} = \begin{bmatrix} x \\ y \end{bmatrix}^{\text{projected}} - \begin{bmatrix} x \\ y \end{bmatrix}^{\text{expected}}. \tag{18}$$

Fig. 8 shows the errors related to the projection of the chessboard corners from the *top left camera* to the *top right camera* before and after the optimization of the position and orientation parameters of the cameras. These results can be better evaluated through the calculated mean error and standard deviation values, as shown in Table 1:

Next, the proposed approach was compared with other calibration methodologies: *stereo calibrate* function[7] provided by OpenCV and the *kalibr* calibration method [26]. The *kalibr* method requires hardware synchronization and receives a bag file as input, unlike the other approaches, which make use of the datasets collected as described in Section 2. Because of this, two different calibrations are provided for *kalibr*: the first in which the training dataset is used, and a second which uses the test dataset, i.e. the dataset which is used to evaluate all approaches.

The results for the proposed approach are presented for two different scenarios, taking into account the camera (top left or top right) which was used for creating the initial values of the chessboard poses (see Eq. (4)). These two variants are used to assess the impact of the selection of the camera for providing initial estimates on the final calibration estimates.

In this experiment, calibration of a pair of sensors composed by the *top left camera* (cam1) and the *top right camera* (cam2) is evaluated. The dataset used for running the calibration procedures, i.e. the training dataset, is composed of 27 collections (27 images per camera). The test dataset to be used to evaluate the estimated sensor-to-sensor transformations has 15 collections. Images from the train and test datasets are similar.

[7] https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html?highlight=stereocalib#cv2.stereoCalibrate.

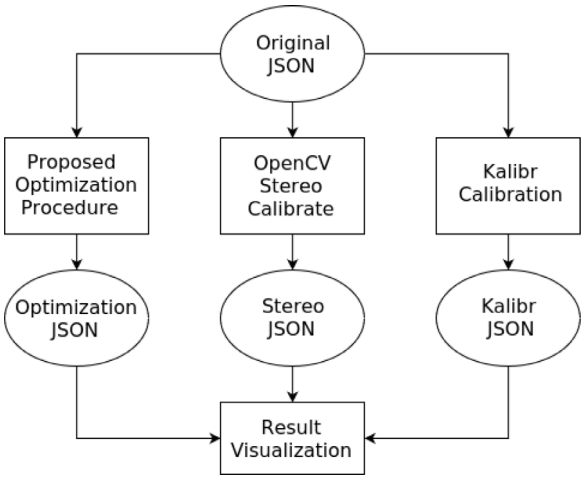


Fig. 9. Flowchart representing the results comparison structure. Each ellipse represents a JSON file and each rectangle identifies a programmed application.

In order to make this comparison fair, the three distinct calibration procedures are given the exact same information. Moreover, the procedures were implemented in such a way that the returned estimated parameters, and remaining data, are organized similarly to the proposed approach. This means that each distinct approach will output a final JSON file with the estimated position and orientation of the sensors. Taking all this into account, a specific tool was created for visualizing the results of the different calibration procedures named *Results Visualization*, which imports the JSON files outputted by each one of the several approaches. Fig. 9 shows a flowchart of this framework, built specifically to compare the proposed methodology with standard pairwise approaches.

Fig. 10 shows the pixel errors of the three distinct calibration approaches. Note that the methodology described above is computed separately for each collection. The performance of the *kalibr* method is clearly below the other two. We suspect there are several factors contributing to this. The first is that this method requires hardware synchronization, not ensured in the used datasets. Another is that the *kalibr* method reads data from a bag file and thus we have no control over the images which are selected to run the calibration. In an attempt to address the problem, we ran a *kalibr* calibration using the test dataset as input (*kalibr* test in Table 2). It may be that the selection of images is not working well, which in turn causes a poor calibration performance. Also, due to the limited duration of the bag file of the experiment, only around 20 to 30 images have been selected, a total similar to the datasets we have used. It could be that *kalibr* requires a larger number of images. In any case, we believe these results are not representative of *kalibr*.

The proposed approach and the stereo calibration display similar errors, which means that the proposed approach is on par with a state of the art calibration approach. Moreover, the largest error of each of the compared methodologies occurs for the same collection (in this case, for collection 4, the dark green). This also shows a high degree of consistency between the proposed approach and the stereo calibration.

Table 2 shows the average error and the standard deviation of all tested calibration approaches.

These results exhibit reprojection errors in the order of some pixels, which is the normal range of values for these methods and experimental setups. Moreover, the obtained values are very similar between the proposed approach and the *stereo calibrate*. As such, results show that the proposed approach is able to calibrate

Table 2
Average errors and standard deviations, in pixels, for the distances in x axis and y axis, for the proposed approach, the OpenCV stereo calibration and the *kalibr* calibration method. For *kalibr*, two datasets were used for training: the train dataset, which was also used to train all other approaches, and the test dataset, which was used to evaluate all approaches. Values in pixels.

Calibration method	Average error		Standard deviation	
	x	y	x	y
Proposed approach (left)	2.218	1.633	1.223	0.584
Proposed approach (right)	2.080	1.797	1.253	0.608
OpenCV stereo calibrate	1.251	0.903	1.509	0.767
Kalibr (train) [26]	67.383	8.887	0.832	1.722
Kalibr (test) [26]	1.187	17.999	1.369	2.225

all sensors on-board the ATLASCAR2 using a single optimization procedure. Furthermore, the accuracy of this joint calibration framework we propose is the same as when using state of the art pairwise calibration methods.

4.2. Complete system calibration

This section will provide results concerning a full system calibration. Note that, in Section 4.1, the results focus only on the evaluation of the camera sensors, despite the fact that the complete system was also calibrated. In this section, the goal is to characterize all the sensors and not just the cameras. Because of this, it is not possible to compare the full system calibration (taking into account all the sensors) with other approaches since, as described in Section 1, there is no calibration framework available, in particular a multi-sensor and multi-modal one.

Fig. 11 shows the average error per sensor over the cost function evaluations, for a full system calibration test. The average error per sensor is estimated after the several error measurements computed for each particular sensor. For example, a camera cost sub-function returns as many residuals as chessboard corners (see Eq. (7)), while the LiDAR sub sub-function returns four measurements (see Eq. (10)). The average error for camera sensors is provided in pixels, while for LiDAR sensors the error is in metres. The first takeaway is that the optimization is working as intended, since the minimization of the errors of all sensors can be observed. This shows that the multi-sensor, multi-modal optimization (the joint minimization of all the sensor's parameters) is in fact possible. Furthermore, the final errors values (after the optimization is finished) are around a few pixels for camera sensors (2.8 and 3.3 pixels for the top left camera and the top right camera, respectively), and around a few centimetres for the LiDARs (0.017 and 0.033 metres for the left laser and right laser, respectively). These values are on par with the state of the art, even when considering calibration results for pairwise approaches. Another important insight is the reason why the top left camera residual starts with a low error: Section 3.1, in particular Eq. (4), described how the initial poses of the chessboards were estimated using one camera sensor, which is arbitrarily selected. In this test, the top left camera was selected to produce the initial chessboard pose estimates. Thus, since the corner detection in the top left camera images are used to compute the initial chessboard poses, the reverse procedure of projecting the chessboard corners back to the image results in corner coordinates that are naturally very close to the detections at the beginning of the optimization.

Fig. 12 shows the data from the LiDARs along with a representation of the chessboard. For a better visualization, a single collection is displayed. The four images correspond to different stages of the optimization process. It is possible to see an improvement during the calibration (i.e. from Fig. 12(*a*) and (*b*), the beginning of the optimization, to (*c*) and (*d*), the end of the optimization, since the data from both LiDARs is much closer

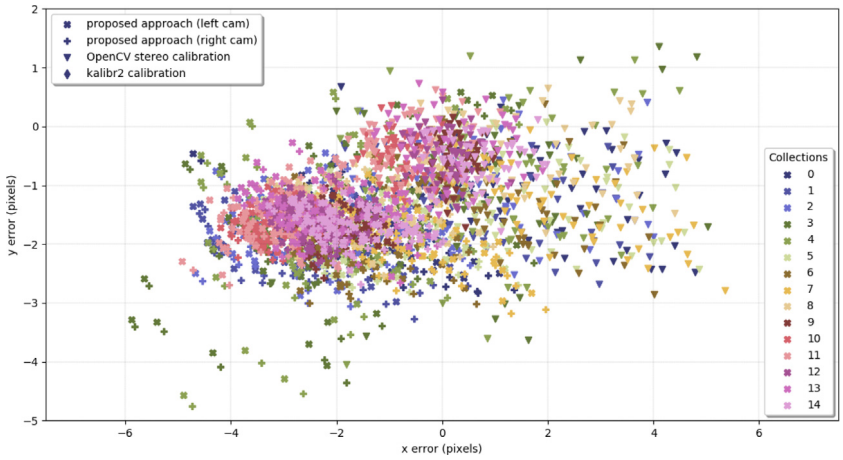


Fig. 10. Pixel coordinate errors between projected and expected chessboard corners. The *kalibr* results are not visible because of the selection of the axes range.

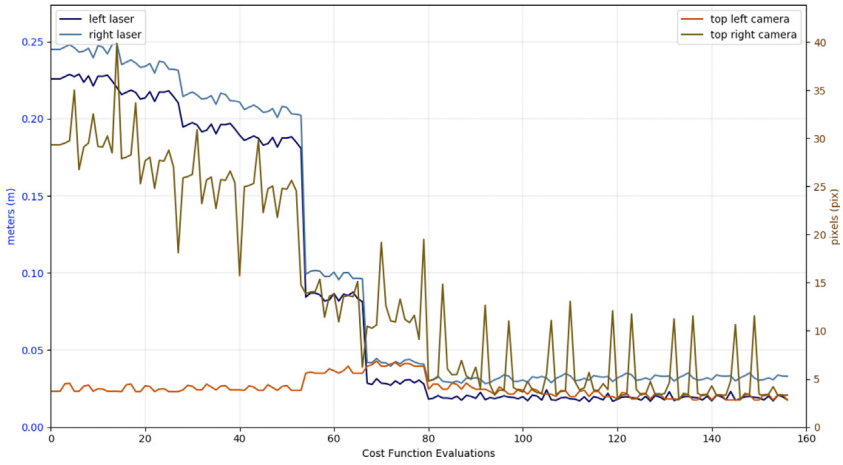


Fig. 11. Average error per sensor during a full system calibration procedure. Errors for cameras in pixels, for LiDARs in metres.

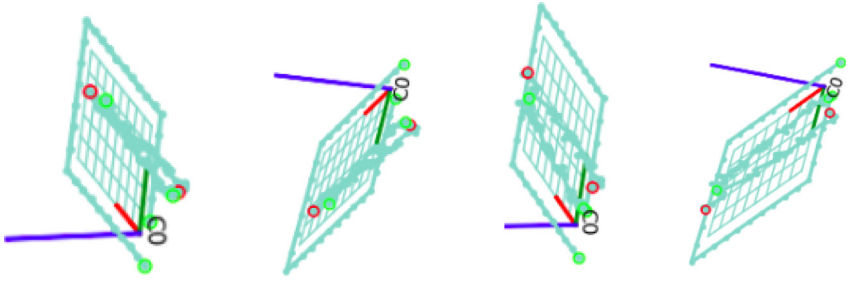


Fig. 12. Left laser (dots surrounded by red circles) and right laser (dots surrounded by green circles) data overlaid onto a representation of the chessboard, taking in consideration the pose of the chessboard and the LiDARs as estimated by the calibration for one particular collection: (*a*) and (*b*) the start of the optimization (initial guess); (*c*) and (*d*) the end of the optimization (calibration results).

to the chessboard plane (*c*) and (*d*) when compared to (*a*) and (*b*). This shows that the proposed approach is also capable of calibrating LiDARs within a joint optimization framework.

5. Conclusions and future work

This paper proposes an extrinsic calibration methodology that is general, in the sense that the number of sensors and their modalities are not restricted. The approach is compliant with the ROS framework, having also the advantage of not altering the *tf* tree. To accomplish this, the problem is formalized as an optimization procedure of a set of partial transformations, which accounts for specific links in the transformation chains of the sensors. Additionally, the work contributes with a set of interactive tools for the positioning of the sensors and labelling of data, which facilitate the creation of a first guess and significantly ease the calibration procedure.

Results show that the proposed approach is able to achieve similar accuracy when compared to state of the art methodologies, implemented in OpenCV. Moreover, these results are

Please cite this article as: M. Oliveira, A. Castro, T. Madeira et al., A ROS framework for the extrinsic calibration of intelligent vehicles: A multi-sensor, multi-modal approach, Robotics and Autonomous Systems (2020) 103558, https://doi.org/10.1016/j.robot.2020.103558.

obtained by performing a complete calibration of the system, rather than one of a single pair of sensors. In other words, the proposed approach calibrates all sensors at once, with similar performance as the pairwise approaches. This confirms that the proposed approach is adequate for the calibration of complex robotic systems, as are most intelligent vehicles.

Future work will focus on the extension to additional sensor modalities, e.g., 3D LiDARs, RGB-D cameras, *Radio Detection And Ranging* (RaDAR), etc. Given the scalability of the proposed framework, it is expected that this should be more or less straightforward. Finally, the ultimate goal is to produce a multi-sensor, multi-modal calibration package that may be released to the community.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

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, as well as CYTED/TICs4CI — Aplicaciones TICS para Ciudades Inteligentes.

References

[1] G.R. Mueller, H. Wuensche, Continuous stereo camera calibration in urban scenarios, in: 2017 IEEE 20th Int. Conf. on Intelligent Transportation Systems (ITSC), 2017, pp. 1–6, http://dx.doi.org/10.1109/ITSC.2017.8317675.

[2] L. Wu, B. Zhu, Binocular stereovision camera calibration, in: 2015 IEEE Int. Conf. on Mechatronics and Automation (ICMA), 2015, pp. 2638–2642, http://dx.doi.org/10.1109/ICMA.2015.7237903.

[3] Rou Su, JingLiang Zhong, QiaoLiang Li, SuWen Qi, HuiSheng Zhang, TianFu Wang, An automatic calibration system for binocular stereo imaging, in: 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conf. (IMCEC), 2016, pp. 896–900, http://dx.doi.org/10.1109/IMCEC.2016.7867340.

[4] Y. Ling, S. Shen, High-precision online markerless stereo extrinsic calibration, in: 2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2016, pp. 1771–1778, http://dx.doi.org/10.1109/IROS.2016.7759283.

[5] V.Q. Dinh, T.P. Nguyen, J.W. Jeon, Rectification using different types of cameras attached to a vehicle, IEEE Trans. Image Process. 28 (2) (2019) 815–826, http://dx.doi.org/10.1109/TIP.2018.2870930.

[6] F. Vasconcelos, J.P. Barreto, U. Nunes, A minimal solution for the extrinsic Calibration of a Camera and a laser-rangefinder, IEEE Trans. Pattern Anal. Mach. Intell. 34 (11) (2012) 2097–2107.

[7] M. Pereira, D. Silva, V. Santos, P. Dias, Self calibration of multiple lidars and cameras on autonomous vehicles, Robot. Auton. Syst. 83 (2016) 326–337.

[8] M. Almeida, P. Dias, M. Oliveira, V. Santos, 3d-2d laser range finder calibration using a conic based geometry shape, in: Image Analysis and Recognition, 2012, pp. 312–319.

[9] A. Geiger, F. Moosmann, O. Car, B. Schuster, Automatic camera and range sensor calibration using a single shot, in: Proc. IEEE Int. Conf. Robot. Autom., 2012, pp. 3936–3943.

[10] C. Guindel, J. Beltrán, D. Martín, F. García, Automatic extrinsic calibration for lidar-stereo vehicle sensor setups, in: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), 2017, pp. 1–6.

[11] Y.C. Kwon, J.W. Jang, O. Choi, Automatic sphere detection for extrinsic calibration of multiple rgbd cameras, in: 2018 18th Int. Conf. on Control, Automation and Systems (ICCAS), 2018, pp. 1451–1454.

[12] A. Khan, G. Aragon-Camarasa, L. Sun, J.P. Siebert, On the calibration of active binocular and rgbd vision systems for dual-arm robots, in: 2016 IEEE Int. Conf. on Robotics and Biomimetics (ROBIO), 2016, pp. 1960–1965, http://dx.doi.org/10.1109/ROBIO.2016.7866616.

[13] F. Basso, E. Menegatti, A. Pretto, Robust intrinsic and extrinsic calibration of rgb-d cameras, IEEE Trans. Robot. 34 (5) (2018) 1315–1332, http://dx.doi.org/10.1109/TRO.2018.2853742.

[14] Y. Qiao, B. Tang, Y. Wang, L. Peng, A new approach to self-calibration of hand-eye vision systems, in: 2013 Int. Conf. on Computational Problem-Solving (ICCP), 2013, pp. 253–256, http://dx.doi.org/10.1109/ICCPS.2013.6893596.

[15] C. Zhang, Z. Zhang, Calibration between depth and color sensors for commodity depth cameras, in: 2011 IEEE Int. Conf. on Multimedia and Expo, 2011, pp. 1–6, http://dx.doi.org/10.1109/ICME.2011.6012191.

[16] G. Chen, G. Cui, Z. Jin, F. Wu, X. Chen, Accurate intrinsic and extrinsic calibration of rgb-d cameras with gp-based depth correction, IEEE Sens. J. 19 (7) (2019) 2685–2694, http://dx.doi.org/10.1109/JSEN.2018.2889805.

[17] Qilong Zhang, R. Pless, Extrinsic calibration of a camera and laser range finder (improves camera calibration), in: 2004 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), Vol. 3, 2004, pp. 2301–2306, http://dx.doi.org/10.1109/IROS.2004.1389752.

[18] M. Häselich, R. Bing, D. Paulus, Calibration of multiple cameras to a 3d laser range finder, in: 2012 IEEE Int. Conf. on Emerging Signal Processing Applications, 2012, pp. 25–28.

[19] Z. Chen, X. Yang, C. Zhang, S. Jiang, Extrinsic calibration of a laser range finder and a camera based on the automatic detection of line feature, in: 2016 9th Int. Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 2016, pp. 448–453.

[20] M. Velas, M. Spanel, Z. Materna, A. Herout, Calibration of rgb camera with velodyne lidar, 2014.

[21] G. Lee, J. Lee, S. Park, Calibration of vlp-16 lidar and multi-view cameras using a ball for 360 degree 3d color map acquisition, in: 2017 IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems (MFI), 2017, pp. 64–69, http://dx.doi.org/10.1109/MFI.2017.8170408.

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

[23] Dezhi Gao, J. Duan, Xining Yang, B. Zheng, A method of spatial calibration for camera and radar, in: 2010 8th World Congress on Intelligent Control and Automation, 2010, pp. 6211–6215, http://dx.doi.org/10.1109/WCICA.2010.5554411.

[24] V. Santos, J. Almeida, E. Ávila, D. Gameiro, M. Oliveira, R. Pascoal, R. Sabino, P. Stein, Atlascar - technologies for a computer assisted driving system, on board a common automobile, in: 13th Int. IEEE Conf. on Intelligent Transpor Tation Systems, 2010, pp. 1421–1427, http://dx.doi.org/10.1109/ITSC.2010.5625031.

[25] Y. Liao, G. Li, Z. Ju, H. Liu, D. Jiang, Joint kinect and multiple external cameras simultaneous calibration, in: 2017 2nd Int. Conf. on Advanced Robotics and Mechatronics (ICARM), 2017, pp. 305–310, http://dx.doi.org/10.1109/ICARM.2017.8273179.

[26] J. Rehder, R. Siegwart, P. Furgale, A general approach to spatiotemporal calibration in multisensor systems, IEEE Trans. Robot. 32 (2) (2016) 383–398, http://dx.doi.org/10.1109/TRO.2016.2529645.

[27] V. Pradeep, K. Konolige, E. Berger, Calibrating a multi-arm multi-sensor robot: A bundle adjustment approach, in: Experimental Robotics: The 12th Int. Symposium on Experimental Robotics, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 211–225, http://dx.doi.org/10.1007/978-3-642-28572-1_15.

[28] M. Oliveira, A. Castro, T. Madeira, P. Dias, V. Santos, A general approach to the extrinsic calibration of intelligent vehicles using ros, in: Robot 2019: Fourth Iberian Robotics Conference, Springer International Publishing, Cham, 2020, pp. 203–215.

[29] G. Bradski, The OpenCV Library, Dr. Dobb's, J. Softw. Tools (2000).

[30] M. Quigley, K. Conley, B.P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A.Y. Ng, Ros: an open-source robot operating system, in: ICRA Workshop on Open Source Software, 2009.

[31] T. Foote, Tf: The transform library, in: 2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA), 2013, pp. 1–6, http://dx.doi.org/10.1109/TePRA.2013.6556373.

[32] J. Hornegger, C. Tomasi, Representation issues in the ml estimation of camera motion, 1999, pp. 640 – 647, http://dx.doi.org/10.1109/ICCV.1999.791285,

[33] S. Agarwal, N. Snavely, S. M. Seitz, R. Szeliski, Bundle Adjustment in the Large, 2010, pp. 29–42, http://dx.doi.org/10.1007/978-3-642-15552-9_3.

Afonso Castro is a junior web developer. He has an M.Sc. Degree in Mechanical Engineering from the Department of Mechanical Engineering of the University of Aveiro (2019). His master specialization was in robotics and, more precisely, sensor calibration. During his M.Sc. Dissertation development, Afonso Castro has published a research article entitled "A General Approach to the Extrinsic Calibration of Intelligent Vehicles Using ROS" for ROBOT2019: Fourth Iberian Robotics Conference, where he has participated and presented the mentioned work.